VISION-BASED LOCALIZATION WITH MAP INFORMATION

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Hang Chu

August 2015

ABSTRACT

Maps are available for various types of environments. Most people can easily read maps and localize themselves. In this thesis we address this problem: Can computer algorithms make use of the map information, and effectively make associations between vision inputs provided by the camera and map clues, to localize the camera? To be specific, we focus on three different scenarios: outdoor localization, vehicle localization, and indoor localization.

BIOGRAPHICAL SKETCH

Hang Chu was born in Shijiazhuang, Hebei, China, on November 22, 1990. He graduated from Shanghai Jiao Tong University, China, with a B.S. in Information Engineering in 2013. He is currently in his 2nd year of study in the Master of Science in Electrical and Computer Engineering program at Cornell University, under the supervision of Prof. Tsuhan Chen.

This document is dedicated to all Cornell graduate students.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Prof. Tsuhan Chen for the continuous support of my study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my graduate study.

Besides my advisor, I would like to thank my minor advisor: Prof. Ashutosh Saxena, for his insightful comments and encouragement, but also for the hard question which incented me to widen my research from various perspectives.

I thank my fellow labmates Dr. Andrew Gallegher, Dr. Amir Sadovnik, Dr. Ruogu Fang, Amandianeze Nwana, Kuan-chuan Peng, and Dong Ki Kim, for the stimulating discussions, and for all the fun we have had in the last two years. In particular, I am grateful to Prof. Noah Snavely and Prof. David Bindel for enlightening me the first glance of computer vision and numerical analysis.

Last but not the least, I would like to thank my family for supporting me spiritually throughout writing this thesis and my my life in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Outdoor Localization: GPS Refinement and Camera Orientation Estimation from a Single Image and a 2D Map

A framework is presented for refining GPS location and estimate the camera orientation using a single urban building image, a 2D city map with building outlines, given a noisy GPS location. We propose to use tilt-invariant vertical building corner edges extracted from the building image. A location-orientation hypothesis, which we call an LOH, is a proposed map location from which an image of building corners would occur at the observed positions of corner edges in the photo. The noisy GPS location is refined and orientation is estimated using the computed LOHs. Experiments show the framework improves GPS accuracy significantly, generally produces reliable orientation estimation, and is computationally efficient.

1.2 Vehicle Localization: Consistent Ground-Plane Mapping - A Case Study Utilizing Low-Cost Sensor Measurements and a Satellite Image

Vision-aided localization systems are often utilized in urban settings to take advantage of the structured environment, the high availability of unique visual features, as well as complimenting aiding measurements from Global Naviga-

tion Satellite System (GNSS). In this paper, we present a case study for roadway texture mapping that combines the low-cost sensor measurements that are already on many production vehicles (e.g. single frequency GPS, wheel odometry, and a forward looking camera) together with a satellite image. The aim of the method presented here is to obtain high resolution texture of the ground plane that is consistent with the low-resolution satellite image through an optimization process that estimates the smooth vehicle trajectory using Maximum-a-Posteriori (MAP). The main benefit of this system comes from the facts that: (1) it utilizes only low-cost sensors and information that are readily available, (2) it can be easily embedded into existing maps. Data and analysis of a drive captured around a block is used in this study.

1.3 Indoor Localization: You Are Here - Mimicking the Human Thinking Process in Reading Floor-Plans

A human can easily find his or her way in an unfamiliar building, by walking around and reading the floor-plan. We try to mimic and automate this human thinking process. More precisely, we introduce a new and useful task of locating an user in the floor-plan, by using only a camera and a floor-plan without any other prior information. We address the problem with a novel matching-localization algorithm that is inspired by human logic. We demonstrate through experiments that our method outperforms state-of-the-art floor-plan-based localization methods by a large margin, while also being highly efficient for real-time applications.

CHAPTER 2

GPS REFINEMENT AND CAMERA ORIENTATION ESTIMATION FROM A SINGLE IMAGE AND A 2D MAP

2.1 Introduction

Urban localization and navigation have become an important application for many mobile phones. To accomplish that, many smartphones have an embedded GPS (Global Positioning System) receiver. However, there are several deficiencies with this localization approach. First, GPS is prone to inaccuracy in several situations, including the urban canyons between buildings in cities. The outdoor accuracy of mobile phone GPS is only 12.5 meters [75]. Secondly, GPS does not indicate the direction that the user is facing, even if perfect localization was achieved.

We address these two problems of GPS by an image-based localization approach. In our approach, we first ask the user to take an image of any nearby building. Then we localize the camera position and solve for the camera orientation, with the layout and structure of buildings from a simple 2D plan view city map. 2D maps are widely available in many cities, generally well maintained and updated, and incorporate sufficient building structural information for our task. In summary, the goal of our work is to *refine the position of a cameras GPS location and estimate the camera pose, based on detecting vertical corner edges from a single cuboid building image and matching these to a 2D city map with building outlines*. Our method does not require the overhead of computing or storing appearance descriptors on buildings or image patches. Instead, we find edges in the image that are likely to exhibit themselves as building corners on a 2D

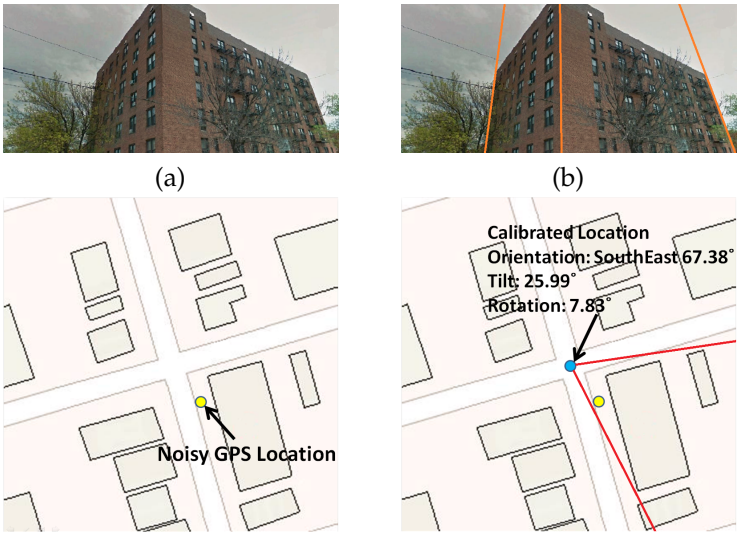


Figure 2.1: (a) The input building image. (b) Identified vertical corner edges of the building. (c) The 2D region map with building outlines and the noisy GPS location. (d) The calibrated location and estimated camera orientation.

map with building outlines. Figure 2.1 illustrates the inputs and outputs of our system.

We conduct experiments on 263 street images collected from 11 unique locations. The results show that our framework is able to accurately perform the task and improve GPS accuracy significantly, suggesting potential applications for mobile localization for tourists. We also test our framework on a dataset of images of apartment buildings.

The contributions of this paper are:

- 1. A framework for refining GPS location and finding camera orientation with a 2D map and a single image.
- 2. The Tilt-Invariant Corner Edge Position (TICEP) feature that is extracted from building images and is useful in computing high accuracy locations.

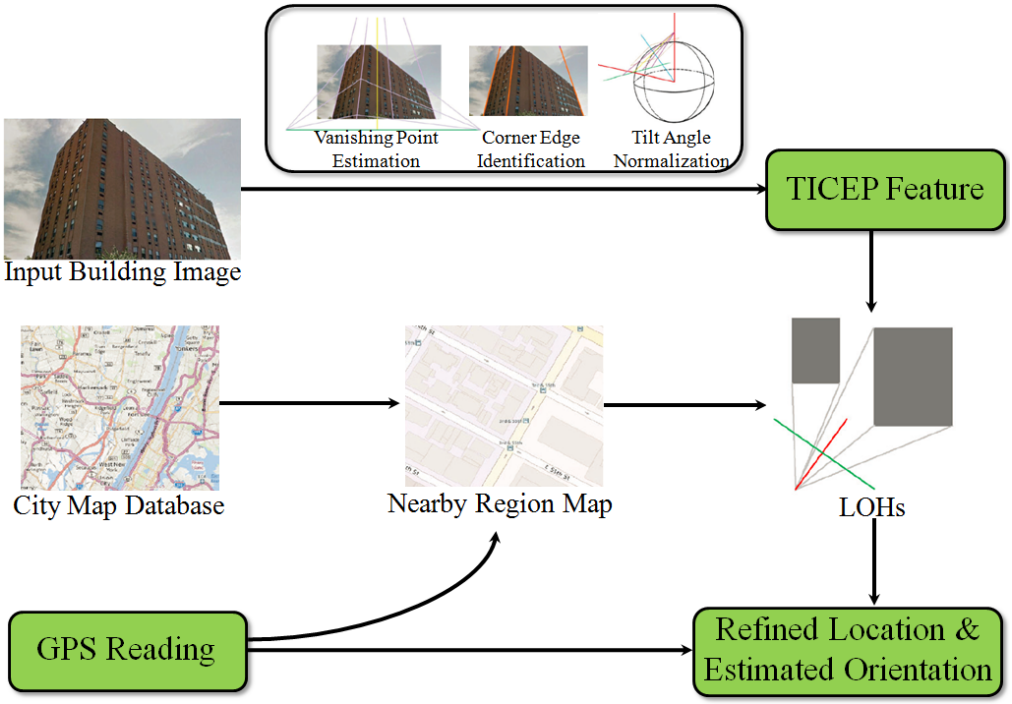


Figure 2.2: An overview of our framework.

- 3. A method for finding Location Orientation Hypotheses (LOHs) that represent possible solutions for camera location and orientation by finding geometric correspondences between corners on the 2D map and extracted TICEP features. From these LOHs, one is selected based on proximity to the initial GPS estimate as the final result.

2.2 Related work

Location-related research has long been an important topic in the computer vision community, perhaps beginning with a challenge to the vision community in 2005 [65]. Hays et al. [29] describes estimating GPS locations from images, using nearest neighbour matching of low- and mid- level appearance features to a large geo-tagged image database. In contrast, our work addresses refining

the geo-position of an image that has an initial GPS estimate.

Another approach is based on the structure-from-motion (SfM) framework, for instance the method Li et al. proposed in [40]. Although their method shows significant accuracy, it requires a huge amount of images and computational resources, essentially requiring that the recognition database be stored in the cloud. Further, there exist difficulties in keeping the reconstructed model up-to-date. In contrast, our approach does not require a database of feature appearances. In a way, our approach can be seen as an extremely simplified variant of SfM, where we put more value on efficiency because the method is seeded with a GPS positional estimate.

In Babound et al. [2], camera pose estimation is performed by finding matches between mountain outlines and a terrain map. In the work of Schindler et al. [59], image lines are used as a geometric feature to construct building models. The success of these methods suggest the effectiveness of interpreting the environment in view of simple structural lines, which inspires us to use vertical corner edges to represent building structures in images.

Park et al. [49] proposed a method of estimating location and orientation of camera by matching the ground view image with a satellite image. Their work can be seen as an intermediate between feature-based matching and symbolic map matching. The work of Ramalingam et al. [56] can be also categorized similarly as in their framework an omni-skyline image is used in an analogous way as the satellite image of [49].

Other relevant work include the work of Chen et al. [11] and Baatz et al. [1]. Though their works are quite different from ours as they mainly considered

image features, they show the effectiveness of using vanishing points and rectification for location recognition in urban area.

The most relevant work is reported in [10] by Cham et al., where vertical and horizontal building edges are extracted from an omnidirectional image comprised of four directional images. From these edges, structural fragments describing a piecewise linear contour of the building are produced and used for searching in a region of 2D map. This inspires us to take a further step: refine the GPS position by discovering structures from a single image and referring to a map. The framework in [10] cannot solve the GPS refinement problem well. First, in [10] the camera tilt (elevation) is not estimated and considered in the localization process. This is fine with large scale block searching and rough localization tasks as shown in [10], but it is problematic to achieve a precision higher than GPS. Second, four directional images are required in their method, which causes more user operation. In our method, we solve the tilt problem by using the TICEP feature, and we need only one image.

2.3 Approach

Our algorithm takes a single building image, a 2D city map with building contours, and a noisy GPS reading as inputs. For the building image, we extract Tilt-Invariant Corner Edge Position (TICEP) features by sequentially applying vanish point estimation, corner edge identification, and tilt angle normalization. Next, we retrieve the nearby region map of the GPS reading from the whole city map database. Using the nearby region map and computed TICEP features, we determine multiple Location Orientation Hypotheses (LOHs) in

the nearby region. Essentially, an LOH is a geographic position and orientation from which a camera could capture a nearby building that will have corners aligning with the observed edges in the image. Finally, the GPS location is refined using the LOHs, and the orientation of the camera is estimated. Figure 2.2 shows an overview.

2.3.1 Computing TICEP features

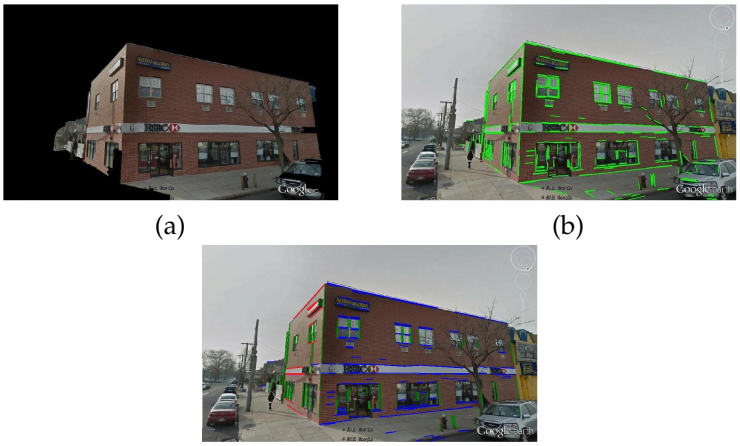
The procedure of computing TICEP features can be divided into three stages: estimating vanishing points, identifying vertical building corner edges, and normalizing the tilt angle.

We first introduce several notations that will be used. For vanishing points (*vp*), we denote the vertical *vp* and the i^{th} horizontal *vp* by $\boldsymbol{v}_v$ and $\boldsymbol{v}_i$. As we are particularly interested in a single Manhattan building, we expect two horizontal vanishing points [13]. We use l_v and l_i to denote line segments labeled to v_v and v_i. I_{ij} is defined to be the set of all intersection points of the extensions of any pair of lines taken from two different horizontal *vp*-labeled line segment sets l_i and l_j, $i \neq j$. S_i denotes one image segment.

Estimating vanishing points

Vanishing points are used for detecting the corner lines, and for estimating the camera focal length. To detect vanishing points of the input image, we perform the following processing steps.

Image segmentation: As in our intended application, we ask the user to take



(a) (b)

(c)

Figure 2.3: (a) An example of selected central segments. (b) An example of detected line segments. (c) An example of *vp* detection and *vp*-labeled line segments, different colors indicate different labels.

an image of any nearby building, it is reasonable to assume that the building is generally centered within the frame of the image. To reduce the occurrence of false vanishing point detections, we first perform a segmentation with the intent of removing non-building segments from the image periphery. To do this, we perform a standard marker controlled watershed segmentation and select segments near the image center, Figure 2.3(a) shows an example.

Line segment detection (LSD): We use the algorithm introduced by Gioi et al. [71] to detect line segments, denoted by l. N_l denotes total number of line segments. Figure 2.3(b) shows an example of LSD.

Vanishing point estimation: We adopt the method in [67] and [79], which are based on the J-Linkage model. Experiments show that this method is highly efficient and accurate for man-made environments [67], which are exactly the properties we want for our algorithm. Figure 2.3(c) shows an example of estimated vanishing points for an image.

Identifying vertical building corner edges

We seek lines in the image that correspond to building corners that will correspond to corners on the building footprint of the 2D map, rather than merely co-planar vertical lines of two facades of the same building.

The identification of building boundary corner edges is now described. We expect that boundary corner edges typically exhibit a large gradient in the horizontal direction, and have different colors (building color and background color) on either side of the boundary. To find these boundary corner edges, we first rectify the input image vertically using the estimated *vp*s, and then we detect image columns that are likely to align with boundary corner edges using a score computed as

$$Scr_{BCE}(j) = Scr_c(j) + Scr_p(j) + Scr_l(j) \tag{2.1}$$

where j is the column coordinate in the vertically rectified trimmed image, Scr_c, Scr_p, and Scr_l are respectively scores of the horizontal pixel color gradient, the number of pixels that change segment label horizontally, and an indication of whether the column contains no horizontal lines and a neighboring column does contain at least one horizontal line. For computing Scr_l, the set of horizontal lines is selected from all horizontal *vp*-labeled line sets that exceed 100 pixels in length. After boundary corner edges are identified, we classify them into left boundary corner edges and right boundary corner edges from the consistency of color, column pixel total number, and coverage of long horizontal lines.

Intersecting corner edges are identified after boundary corner edges. In-

Figure 2.4: An example of identified vertical corner edges, blue indicated boundary corner edges, red indicates an intersecting corner edge, and green stands for the horizon.

spired by [10], we define intersecting corner edges as a vertical *vp*-labeled line segment that intersects with horizontal line segments belonging to 2 different *vp*s. In addition, we require that only one intersecting corner edge exists between a left boundary corner edge and a right boundary corner edge. The longest line is selected if more than one candidate is found. Figure 2.4 shows an example of identified vertical corner edges. We then compute the corner positions as the intersections of identified corner edge and the horizon (purple circles in Figure 2.4). Denote the i^{th} corner edge and corner edge position as $\boldsymbol{e}_i$ and $\boldsymbol{p}_i$, we have $\boldsymbol{p}_i = \boldsymbol{e}_i \times (\boldsymbol{v}_1 \times \boldsymbol{v}_2), i = 1, , N_p$.

It should be noticed that as real world images are more complex than ideal examples, sometimes our algorithm fails to identify all the corner edges correctly. However, with the vanishing points that can be estimated by our algorithm accurately, the user can manually correct an identification result by a single tap. We also show in Section 4 that our algorithm is able to identify corner edges with 85.73% accuracy, which means that our algorithm can generally give satisfactory identification results automatically and large amount of user effort can be saved.

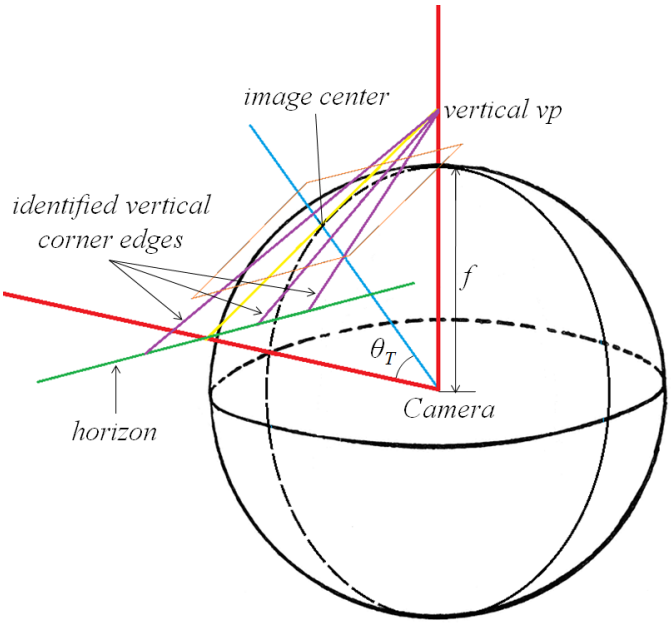


Figure 2.5: Diagram of computing camera tilt angle (with rotation angle rectified).

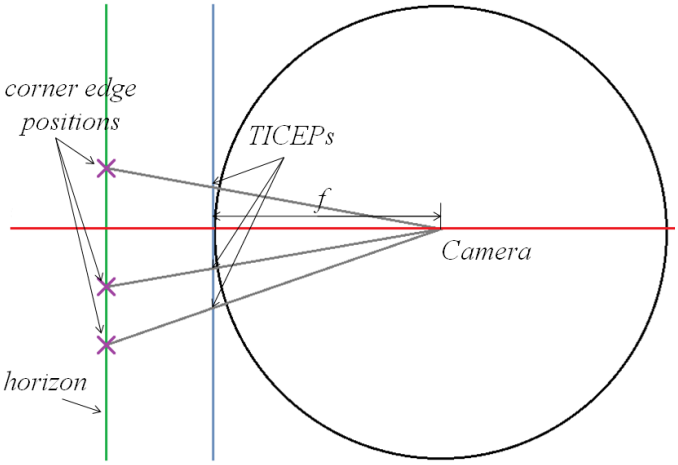


Figure 2.6: Diagram of normalizing the tilt angle. This is Figure 2.5 looking from top to bottom.

Normalizing the tilt angle

The computed corner edge positions of a image are variant to the camera tilt angle. We now describe the normalization of tilt angle. The first step is to estimate several camera parameters: rotation angle θ_R, focal length f, and tilt angle θ_T.

We first compute the rotation angle using a procedure similar to the method in [24]. Then we rectify the coordinates of all *vp*s and intersecting points according to the computed rotation angle, so that the vertical *vp* $\boldsymbol{v}_v$ now lies on the images *y*-axis, and the two horizontal *vp*s $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ lie on a line parallel to the *x*-axis. Also all the intersecting points now have the same *y*-coordinate.

After we neutralize the camera rotation, the focal length can be easily obtained as [8]

$$f = \sqrt{-v_{vy}v_{1y}} \tag{2.2}$$

It should be noted that when image has no tilt we have $v_{vy} = -\infty$, the focal length cannot be computed and must be obtained from the image format file. As shown in Figure 2.5, the tilt angle of the camera can be computed as

$$tan(\theta_T) = \begin{cases} \frac{h-\sqrt{h^2-4f^2}}{2f} & \text{when } |v_{vy}| > |v_{1y}| \\ \frac{h+\sqrt{h^2-4f^2}}{2f} & \text{when } |v_{vy}| \leq |v_{1y}| \\ 0 & \text{when } v_{vy} = -\infty \end{cases} \tag{2.3}$$

where

$$h = |v_{vy}| + |v_{1y}| \tag{2.4}$$

As shown in Figure 2.5, the edge positions p_i are affected by the tilt angle, thus we normalize the tilt angle and compute the TICEP as

$$TICEP_i = p_{ix}\cos(\theta_T) \tag{2.5}$$

Figure 2.6 shows a diagram for the tilt angle normalization. We only care about the horizontal coordinates of the intersecting points as they are sufficient to demonstrate the distribution of corner edges.

Now that we have finished all procedures related to the building image, and an image is represented as the focal length and TICEPs, i.e., $Image = \{f, \boldsymbol{TICEP}\}$.

Except for the orientation angle θ_O, all the parameters of camera pose have been estimated. θ_O will be estimated in Section 3.2 together with the location.

2.3.2 Refining GPS by LOH

For the entire city map, we first extract the region map as the *200m-by-200m* square region centered at the noisy GPS location. The range of the region map is generally far larger than the noise of GPS so that the correct location is included in the region map with high confidence.

We now describe the computing of an LOH. Assume we know the correspondence of the computed $\boldsymbol{TICEP} = \{TICEP_i\}$ to the corners in the map. The locations of the corresponding map corners are $\boldsymbol{c}_h = \{\boldsymbol{c}_{hi}\}$. An LOH is defined as the particular location and orientation in the map, from where the corners in c_h can be seen in the way that $\boldsymbol{TICEP}$ are distributed. To describe an LOH, its location and orientation are needed: $LOH = (\boldsymbol{x}_{LOH}, \boldsymbol{n}_{LOH})$, where $\boldsymbol{x}_{LOH}$ is the position on the map, and $\boldsymbol{n}_{LOH}$ is a normalized orientation vector.

To compute the parameters of an LOH associated with corners of a building footprint, we minimize the total deviation between the positions where corners

would be seen in the image plane from the view of a potential LOH and TICEPs, i.e., the summation of distances between the intersection of LOH-corner line and image plane and location of the corresponding corner edges on the image plane:

$$(\boldsymbol{x}_{LOH}, \boldsymbol{n}_{LOH}) = \underset{(\boldsymbol{x},\boldsymbol{n})}{\arg\min} \sum_{1 \le i \le N_p} ||\boldsymbol{q}_i - \boldsymbol{inter}_i||^2 \tag{2.6}$$

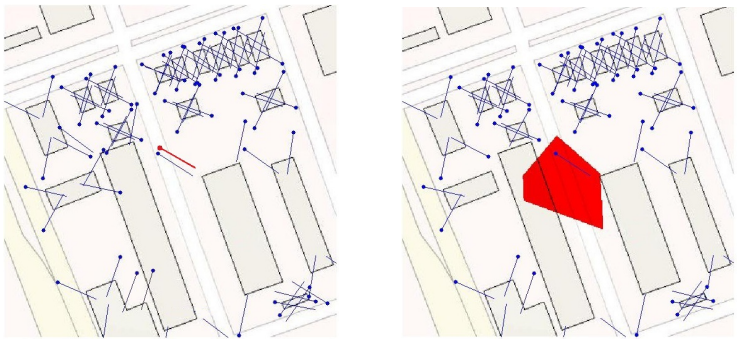
where

$$\boldsymbol{inter}_i = (\boldsymbol{x} \times \boldsymbol{c}_{hi}) \times (\boldsymbol{q}_1 \times \boldsymbol{q}_2) \tag{2.7}$$

$$\boldsymbol{q}_i = \boldsymbol{x} + f\boldsymbol{n} + TICEP_i \boldsymbol{n}_\perp \tag{2.8}$$

The minimization problem is not linear. As one LOH has three degree of freedom, when $N_p = 3$ a precise multinomial approximation can be found for the two coordinates and one orientation angle by taking a Taylor expansion and solving closed-form equations. When $N_p > 3$, as we already have an efficient solution for three corners, RANSAC [20] could be applied to solve the problem, although this will be implemented in the future. At present, all possible sets of three adjacent corners from each building outline are selected to solve for a candidate LOH. We find, in general, the strategy of using 3 corners is reliable, and results are given in Section 4. Figure 2.7(a) shows an example of the set of candidate LOHs that our algorithm finds.

From Figure 2.7(a), it can be seen that not all LOHs are reasonable: some LOHs are indoor, some LOHs have their visibility blocked by another building so they are not able to have visual of the corners they are matching with. To eliminate those bad LOHs, we perform a visibility check. We exam the visibility



(a) (b)

Figure 2.7: (a) An example of solved LOHs (blue), correct location and orientation (red). (b) Example of visible LOHs, and area when GPS falls in the correctly corresponded LOH can be found.

of a LOH by checking if its sightlines to the matched corners intersect with any building walls, and, if so, that LOH is eliminated.

As the last step, we take the visible LOH that is nearest to the noisy GPS location. In practice we find the correctly corresponded LOH is often selected when the noise of GPS is not too large. A major cause of deviation from the correctly corresponded LOH to the correct location is the accuracy of vertical corner edge positions in the building image, and that accuracy can be achieved with fairly small error. Thus our algorithm is able to produce results where the correctly corresponded LOH deviates from the correct location by only a few meters. Figure 2.7(b) shows visible LOHs and the red color indicates the area that if the noisy GPS falls within it, the correctly corresponded LOH will be selected.

2.4 Experiments

To evaluate our framework, we first collect 390 images using Google Street View from 11 unique locations in New York City to simulate user input images. We apply our TICEP feature extraction procedure on each image. We define a successful detection as all detection results deviate less than 20 pixels from the ground truth. Our algorithm identifies 1003 corner edges successfully out of all 1170 corner edges (85.73%), also in 263 images (67.44%) all the edges are correctly identified. It should be noted that we detect three corner edges in each image, in practical applications the detection can be improved significantly with very little user aid. To measure the performance of GPS refinement and orientation estimation using TICEP+LOH, we use the 263 images with successful detection for the next experiment. The 2D region maps with building outlines are collected from *here.com*. We implement the framework using a mixture of c and Matlab, all the experiments are tested on an Intel Core i5 2.40GHz PC. The average run time of our whole framework is 1.7 seconds per image. Finally the dataset and code are available at *chuhang.github.io/vision.html*.

For each image and its corresponding region map, we first test the location and orientation error of the correctly corresponded LOH (i.e., the LOH computed with correct correspondence between building map corners and detected TICEPs) to the ground truth location and orientation provided by Google Street View. We compare our method with the method using the VCLH feature in [10] (corner edge positions without considering the influence of tilt angle) instead of our proposed TICEP feature. We compute the Root Mean Square Error (RMSE) of all 263 images, as listed in Table 2.1. Our method outperforms the compared method in both location and orientation. The RMSE of location of the correct

Table 2.1: RMSE of location and orientation of the correctly corresponded LOH of different methods.

	Proposed Method	Using VCLH in [10]
Location	$2.48m$	$18.68m$
Orientation	$1.6°$	$5.9°$

LOH in our method is significantly smaller than the accuracy of a common mobile phone GPS in outdoor urban area (12.5 meters according to [75]). That explains why our method is able to improve the accuracy of GPS. In the first and second row of Figure 2.9, we show some examples of this experiment.

We have shown why our method is able to refine GPS, we now conduct the main experiment where we solve all LOHs among the map and combine our method with simulated noisy GPS to find the final refined location and orientation. To simulate the noise of GPS, we use a gaussian distributed noise with different standard deviations as suggested in [75]. We also compare our method with the method using VCLH feature in [10]. We experiment with different GPS noise standard deviation σ and 2000 noisy GPS locations are simulated for each image, so for each value σ we have 263×2000 test samples. Figure 2.8 shows the results.

Figure 2.8(a) show that when GPS has low noise, doing nothing (i.e., using the GPS estimate) produces the most accurate estimate. When the noise of GPS becomes larger, our method begins to outperform pure GPS by refining the noisy GPS to the correctly corresponding LOH. The average error of our method also increases as GPS noise increases, because as GPS uncertainty increases, it is more likely that the wrong LOH is closest to the GPS estimate (see the third row of Figure 2.9). This is also described in Figure 2.8(c). When using the VCLH feature in [10] where the influence of tilt angle is omitted, performance degrades

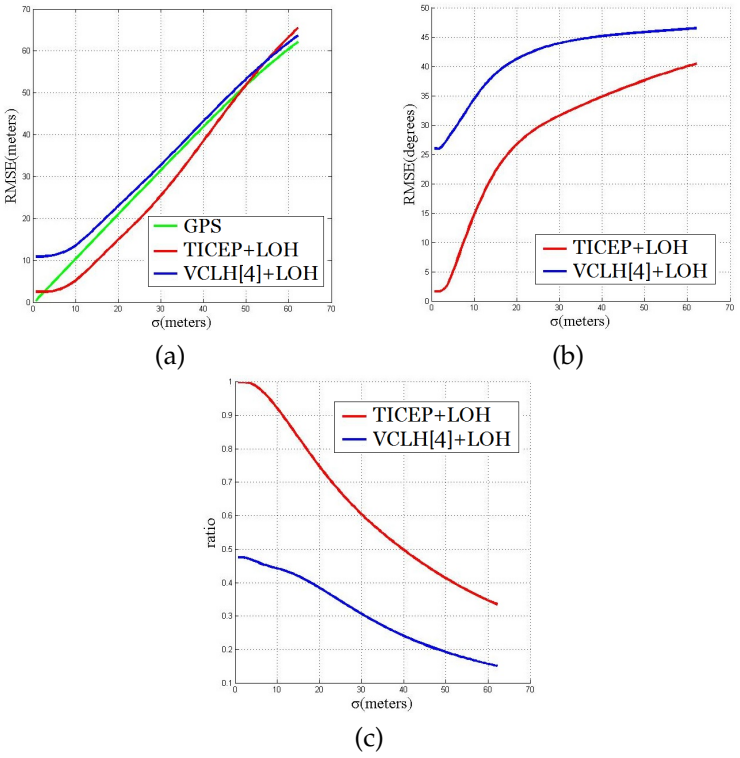


Figure 2.8: Localization and orientation estimation results of different methods. (a) shows location error. (b) shows orientation error. (c) shows the ratio of samples where the correctly corresponded LOH is selected. σ measures the noise of GPS.

and an accuracy that is higher than pure GPS cannot be achieved (as shown in Figure 2.8(a)). According to [75], the general RMSE of mobile phone GPS is 12.5 meters. Under such a noisy condition, our method is able to reduce the RMSE to 6.89 meters and the orientation estimate has average error $17.96°$.

Figure 2.8(b) also indicates a drawback of both methods. When GPS is accurate, the correctly corresponded LOH is selected so the orientation estimation is fairly accurate. However, in cases such as the first and third column of Figure 2.9, due to the layout of buildings, the correct location can be near to the border of the refinable area. That means there exists an incorrectly corresponded LOH near the correct location. This does not bring too much trouble to location refinement because that incorrectly corresponded LOH is not far from the cor-

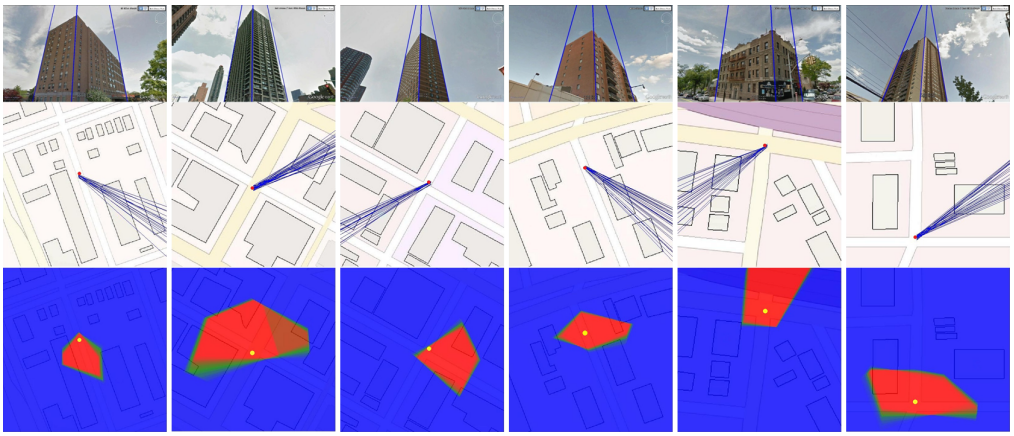


Figure 2.9: Top row: example building images with identified vertical corner edges. Middle row: 2D map with correctly corresponded LOH of images from same location (blue), ground truth location (red). Different images from the same location vary in ground truth orientation. Bottom row: The yellow dot shows the ground truth location. The red pixels show the locations to which the nearest LOH is the correctly corresponded LOH, so when noisy GPS falls in the red area the correctly corresponded LOH is selected (which we term as *refinable area*). These maps are averaged across all images of the same location.

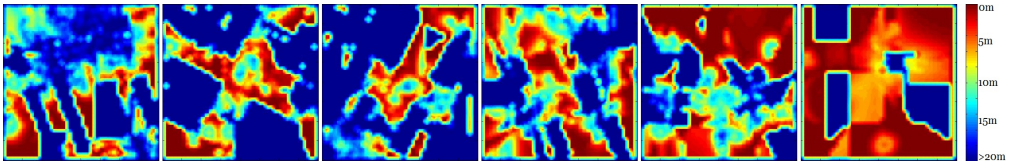


Figure 2.10: Heat maps of RMSE for every possible location on the map, sampled every 5 meters. We consider only outdoor locations. Pure blue means this location is either an indoor location or a location without any building (in any direction) that has at least three visible corners. When the user is at the blue area our method does not work, when the user is at the red area our method works well and produces small RMSE for location. The heat maps correspond to the maps in Figure 2.9.

rect location, but the estimated orientation can be affected significantly because incorrect corner matches are used. It should be noted that this problem is unavoidable unless increase the number of corner edges that are considered or use other features.

To further measure the generality of our framework, we conduct another experiment that estimates an upper bound on the performance of our method. For

Figure 2.11: Left: Building images and their correctly corresponded LOHs in the 2D map, numbers show the RMSE of location and orientation. Right: Images, refinable areas (red), 38%, 68%, 95% of noisy GPS samples (concentric circles).

every outdoor location in the region map, we assume we have a pseudo image taken at that location of the nearest building with at least 3 visible corner edges, and measure the location RMSE of our refined result using 2000 simulated noisy GPS readings from a $12.5m$ gaussian noise distribution. In Figure 2.10 we show heat maps of RMSE. Figure 2.12 shows the distribution of RMSE errors for location estimation. For all outdoor locations of all our collected region maps, the percentage of area with at least one building with three visible corners is 80.6%, and the mean RMSE for all qualified locations is 6.70 meters. This indicates that for a large portion of urban environment, our method can be applied to refine noisy GPS location.

As the last experiment and a full demonstration of our intended application, we take building images in an apartment area using a mobile phone. Then we apply our framework with user aid in correcting mistakes in the corner edge identification step. We simulate 2000 noisy GPS locations for each image using a gaussian distribution with the standard 12.5 meter RMSE. Figure 2.11 shows the results and Table 2.2 lists statistics.

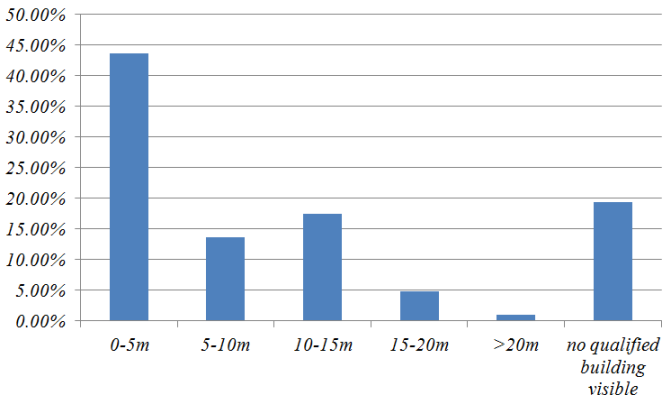


Figure 2.12: Histogram of RMSE of all outdoor locations.

Table 2.2: Statistics for the last experiment.

% of selecting correctly corresponded LOH	Location RMSE	Orientation RMSE
71.52%	$6.55m$	$12.7°$

2.5 Conclusion

We have presented a framework for refining a noisy GPS location and estimating the camera orientation using a building image, and a 2D map. We extract Tilt-Invariant Corner Edge Position features from the image, and identify plausible camera locations and orientations on the map that would result in images having lines at the observed positions. A set of Location-Orientation Hypotheses are proposed to describe the interaction between extracted features and the map effectively. Experiments show that our framework is able to improve accuracy of GPS, and determine the cameras orientation on the map. This framework could be useful for tourist navigation in an urban environment.

CHAPTER 3

CONSISTENT GROUND-PLANE MAPPING: A CASE STUDY UTILIZING LOW-COST SENSOR MEASUREMENTS AND A SATELLITE IMAGE

3.1 Introduction

Autonomous vehicle has become an important topic in both robotics and intelligent transportation communities in recent years. Accurate estimation of the vehicle state is amongst one of the core problems that needs to be solved. Knowledge of the driving environment is essential for a robotic vehicle to accurately recover its state, hence successfully carrying out desired tasks such as complying with traffic rules and ensuring driving safety.

Satellite (aerial) imagery is an useful resource for providing driving environment information. It has advantages in two aspects. Firstly, taking images from high altitude observing point naturally yields high global consistency. Secondly, satellite image databases are well maintained and easy to gain access (e.g. Google maps, Bing maps, etc.). However, satellite images, especially the publicly available ones, suffer from the problem of limited resolution. This makes them less desirable to be directly used in advanced driver assistance systems. Current mainstream precise digital map generation systems provide maps in high quality, but they usually rely on expensive and dedicated sensor set as well as great efforts in manual data analysis. These properties limit their benefits to normal commercial cars.

We present a study for high resolution lane image generation. We use publicly available low resolution satellite images and widely installed low-cost sen-

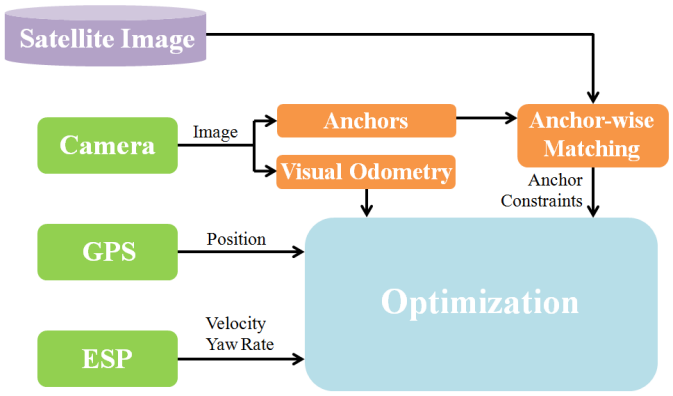


Figure 3.1: An overview of our framework.

sors such as single frequency GPS, wheel odometry computed in the Electronic Stability Program (ESP) system, and a forward looking camera. Our method produces high quality lane images that are consistent with the satellite image, that can be easily embedded into existing maps. The use of low-cost sensors allows standard vehicles to serve as probe cars, which provides fast and inexpensive coverage of large accessible areas as well as up-to-date information.

Figure 3.1 shows the framework overview of our method. We propose to identify anchors and match with the satellite image in the unit of anchor. An anchor is a set of consecutive frames that have highly identifiable structured features. Anchors are identified using orthographic views obtained by the camera. Only identified anchors are matched with the satellite image to compute anchor-wise location constraints. Vehicle states are optimized by minimizing the sum of the weighted residuals in: positions measured by GPS, velocity and yaw rate measured by ESP, visual odometry, and the computed anchor constraints. The anchor-based scheme works more effectively than directly correlating camera and satellite images for each single frame due to the high uncertainty in single frame matching.

3.2 Related Work

Previous work have shown that a great amount of effort is needed to generate high-precision digital maps. However most existing map generation systems commonly require specialized sensors [58, 72] and intense manual analysis [3]. This makes them difficult to be applied in large scale and maintain up-to-date. In another category of work [51], digital maps are generated using satellite images exclusively. Such method has the disadvantages of expensive data acquisition and unable to recover details that can only be observed in close distance.

In recent years road orthographic image construction systems using only commodity sensors have been developed. The methods in [25] and [48] use vision sensor exclusively. Though visual odometry works fully automatically and produces generally accurate local state estimation, it relies heavily on visual cues and does not work well when insufficient texture is observed in the image. Furthermore, vision exclusive methods suffer from accumulated errors, which reduces global consistency by a large margin. The method in [44] uses visual odometry in complement to GPS. However it fails to create road images that are in pixel-level alignment with the satellite image due to limited precision of GPS.

In the work of [46], GPS and Inertial Navigation System (INS) sensors are coupled and then used for absolute localization. In its recent successor [6], visual odometry and road network topologies are also added to improve localization accuracy. These methods show promising results as well as state-of-the-art accuracy. However, as GPS may suffer from insufficient number of visible satellites and multi-path reflections, it is essentially unable to guarantee high absolute positioning accuracy. Besides, though road network structure information

is used in [27] to improve accuracy, sections between road segment intersections are not precisely constrained and thus not accurately mapped.

Methods in [36], [60], and [50] create maps that are consistent with satellite image priors. The satellite image is matched with 3D range scans of outdoor building structures in [36] and [50], and in [60] matched with reconstructed 3D point clouds of the road plane. Although these methods ensure high consistency with the satellite image, the usage of 3D range scanner or stereo 3D reconstruction systems significantly increases the implementation cost and decreases the number of probe vehicles. The method in [45] uses only low-cost sensors along with pre-existing map priors. However it requires labeled digital lane marks, which requires manual efforts and limits the method's usability in large scale.

The main novelties and contributions of this paper are: (1) a study that uses conventional low-cost sensors to create high resolution roadway images that are consistent with the publicly available satellite image, and (2) improved global consistency by identifying anchor points that are common in both the forward looking camera and the satellite image.

3.3 Approach

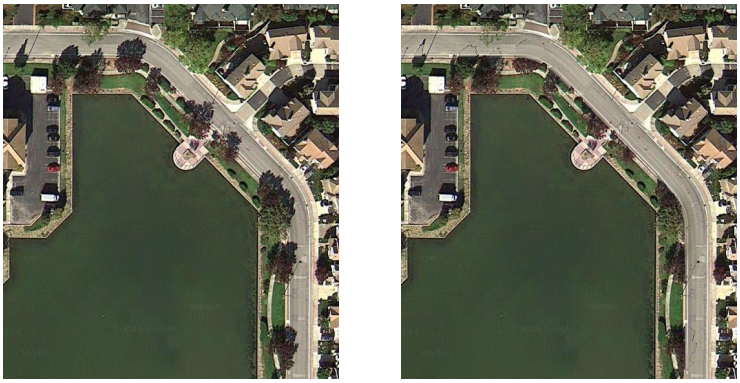
The method includes three major components: a preprocessing preparation stage, an anchor-based method for consistent roadway high resolution image generation, and an incremental optimization algorithm that is more computationally efficient.

3.3.1 Preprocessing

Two preprocessing steps are carried out. The first one is to compute the GPS-to-image projection. The projection is defined as a homography transformation that transforms a *longitude-latitude* GPS coordinate into its corresponding *x-y* image coordinate, where we assume no distortion in the satellite image. To do this, we survey GPS positions along with image coordinates of 20 points randomly scattered in a $1.5km \times 1.5km$ region. The homography matrix is then computed using Random Sample Consensus (RANSAC).

The second preprocessing is detecting and removing shadows in the satellite image. Tree or building often project shadows on the lane, which interferes matching camera images because shadow position varies in different time of the day. To detect shadows, we apply watershed segmentation on the image region within a constant distance to the road centerline. Segments with dark color appearances are treated as shadows. We remove the detected shadows by linear interpolation along the road direction. Figure 3.2(a) shows an example of the original satellite image, Figure 3.2(b) shows the processed image after shadow removal.

It should be noted that both preprocessing steps can be easily automated and thus will not impose requirements for extra manual analysis.



(a) (b)

Figure 3.2: An example of the shadow removing preprocessing, (a) original satellite image, (b) processed image.

3.3.2 Anchor-based Method for Consistent Roadway Image Generation

The orthographic views are obtained by applying homography transformation on a fixed region in the forward-looking (zero-tilt) images. We compute edges from the preprocessed satellite image and orthographic view inputs using the random-forest-based edge detection method [17], which is robust to illumination changes and computationally efficient. For an initial state, edge maps of transformed orthographic camera input and the nearby satellite image region centered at initial location are computed and matched. The matching correlation is used to compute the optimal position the current frame should be shifted to. This aligns the orthographic view with the satellite image. This is a nontrivial task because optimizing directly using computed correlations of all single frames will not work. Figure 3.3 shows examples of orthographic camera inputs. Figure 3.3(a) and Figure 3.3(b) show clear lane marks. These images often provide useful matching results, e.g. Figure 3.3(a) provides accurate match result in the direction perpendicular to vehicle heading; Figure 3.3(b) provides

useful positioning information in both vehicle heading and perpendicular directions. However, not all observed orthographic views contain meaningful textures, such as Figure 3.3(c). In that case, matching fails to provide useful alignment information, sometimes even shows strong correlation at wrong locations.

We further demonstrate why directly matching in single frames fails to produce accurate results by the example in Figure 3.4. Figure 3.4 shows the estimated Δx (the optimal adjust distance in the direction perpendicular to vehicle heading) of different methods from an actual data sequence. At frame 80-200, as input images are textureless similar to Figure 3.3(c), single frame matching shows strong correlation at wrong locations. To address this problem, we propose to use an anchor-based method. An anchor consists of a set of continuous anchor frames, anchor frames are frames that have highly identifiable structured textures. In the example of Figure 3.4 two anchors are found. Frames within an anchor are adjusted uniformly based on all single frame matching correlations. In this way, our method achieves three benefits: (1) it automatically turns off unhelpful matching results; (2) it computes more accurate position constraints by taking into account all correlations within each anchor; (3) it produces smooth adjustment by moving anchors uniformly, which improves final mapping quality.

In the next, we will describe the method we use for anchor identification, visual odometry, obtaining the optimal state estimation by incorporating anchor constraints and all other measurements , as well as the final image generation using estimated states.

(a) (b) (c)

Figure 3.3: Examples of transformed road orthographic views. When matched with the satellite image (a) and (b) produce useful location information. In the contrast images like (c) often fail to provide helpful positioning information.

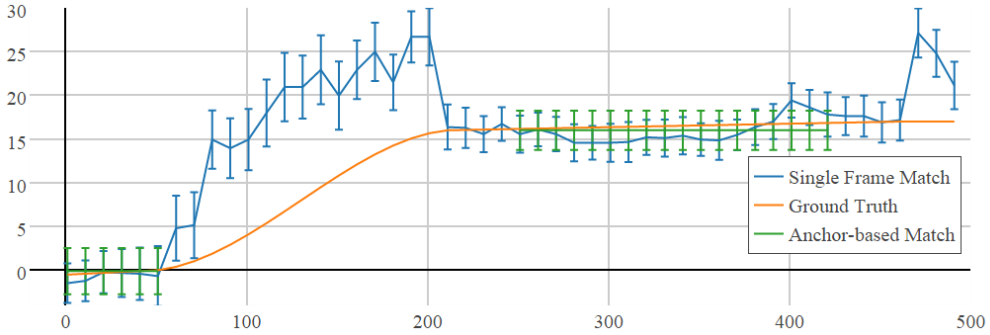


Figure 3.4: An illustration of different matching methods. *x-axis*: time as frame number, *y-axis*: adjustment Δx produced by image matching, bars show standard deviations. The anchor-based method automatically turns off bad matching results, also produces smoother and more accurate location constraints than single frame matching within turned on anchor sections.

Anchor Identification

In this step, the goal is to find whether an orthographic view contains identifiable texture in vertical (vehicle heading or image y) or horizontal (perpendicular to vehicle heading or image x) directions. First, we sum up edge potentials of the orthographic view along two axes into two 1D signals, i.e.

Table 3.1: List of features used in anchor frame identification.

Feature	Definition
mean	$f_1 = \mu\{E_{1D}\}$
variance	$f_2 = \sigma^2\{E_{1D}\}$
peak value	$f_3 = max\{E_{1D}\}$
peak-mean ratio	$f_4 = f_3/f_1$
3 dimensional histogram	$f_5 = 1\{E_{1D} \geq 0.7f_1\}$ $f_6 = 1\{0.3f_1 < E_{1D} < 0.7f_1\}$ $f_7 = 1\{E_{1D} \leq 0.3f_1\}$

$$Ex_j = \sum_{i=1}^{h} E_{ij} \tag{3.1}$$

$$Ey_i = \sum_{j=1}^{w} E_{ij} \tag{3.2}$$

where E_{ij} is edge potential of the orthographic view, h,w are image height and width. To identify anchor frames, we define and extract 7 features from the 1D edge signal. Table 3.1 lists the features, where E_{1D} can be either Ex or Ey.

We collect two anchor/non-anchor training sets corresponding to the two axes respectively. Then a standard *k*-nearest neighbours (*k*-NN) classifier is applied using the normalized feature vector to determine whether an input should be treated as an anchor frame. We further apply a round-robin buffering scheme to filter the computed labels. This process rejects lone outliers and ensures the continuity of identified anchor sections. Each set of continuous anchor frames forms an anchor, anchor constraints are then computed by averaging matching correlations within anchors.

As road images show limited number of repetitive patterns, it is possible to collect a sufficient dataset that can be used for anchor identification in larger

areas. However, such large scale experiment is beyond the scope of this study.

Visual Odometry

As we already have ESP that measures vehicle motion, we want the visual odometry process to produce motions that has greater, or at least comparable accuracy to ESP. However, most traditional image feature based methods fail to meet this requirement. Thus we use a pixel-level dense stereo optimization approach to achieve high-precision visual odometry.

We follow the method introduced in [27] and [62]. The lower half of the transformed orthographic view is used as the region-of-interest (ROI) to optimize the 2D vehicle motion $\boldsymbol{\beta}_{vo} = (\Delta x_{vo}, \Delta y_{vo}, \Delta \alpha_{vo})^T$, where $\Delta x_{vo}, \Delta y_{vo}, \Delta \alpha_{vo}$ are the estimated vehicle translations and yaw angle change. The optimal motion vector is obtained as

$$\hat{\boldsymbol{\beta}}_{vo}^{t} = \arg\min_{\boldsymbol{\beta}_{vo}} \sum_{p \in ROI} \sum_{k} \lambda_k (\boldsymbol{\Phi}_k^{t+1}(p) - \boldsymbol{\Phi}_k^{t}(p')) \tag{3.3}$$

where p' denotes the pixel corresponding to p under the current motion $\boldsymbol{\beta}_{vo}$. $\boldsymbol{\Phi}_k^t$ denotes the k-th feature at frame t, with weight λ_k. We use three feature types: image intensity values, edge potential values, and distance transform values of the edge map. A Nelder-Mead simplex routine is used for minimizing the objective function. Also linear interpolation is used for non-integer pixel coordinates. We use the ESP measurement as initial estimation of $\boldsymbol{\beta}_{vo}$.

State Estimation

At this point, we have derived anchor constraints by matching satellite and camera images at anchor frames, visual odometry derived by matching two consecutive frames, along with GPS and ESP readings. GPS provides noisy absolute location measurement, and it is refined by anchor constraints. Similarly, ESP provides initial relative motion measurement, and it is refined by visual odometry. All measurements and constraints are then used for the final optimization.

The optimization starts with an initial state estimate using an Extended Kalman Filter (EKF) on GPS aiding odometry measurements. We then identify anchors using only input images. For a frame inside an anchor, we match its (Ex, Ey) with (Ex', Ey') that are computed similarly but using the satellite image patch centring and oriented at the initial state. Matching correlations are averaged within each anchor to obtain the unified anchor constraints, which gives anchor-wise optimal adjustments and covariances. Note that here we assume error propagation inside anchors is negligible.

The optimal trajectory $\boldsymbol{s}^{1:M}$ can be computed iteratively as a nonlinear weighted least squares problem as shown in [16] and [72]. The objective function is defined as

$$
\begin{aligned}
E(\boldsymbol{s}^{1:M}) = & \sum_{i=1}^{M-1} (\boldsymbol{p}_{esp}^{i}{}^{T} \boldsymbol{C}_{esp}{}^{-1} \boldsymbol{p}_{esp}^{i} + \boldsymbol{p}_{vo}^{i}{}^{T} \boldsymbol{C}_{vo}^{i}{}^{-1} \boldsymbol{p}_{vo}^{i}) \\
& + \sum_{i=1}^{N} \boldsymbol{p}_{gps}^{i}{}^{T} \boldsymbol{C}_{gps}^{i}{}^{-1} \boldsymbol{p}_{gps}^{i} \\
& + \sum_{i=1}^{K} \sum_{j=1}^{L_i} \boldsymbol{p}_{anchor}^{ij}{}^{T} \boldsymbol{C}_{anchor}^{i}{}^{-1} \boldsymbol{p}_{anchor}^{ij}
\end{aligned} \tag{3.4}
$$

$$
\boldsymbol{p}_{esp}^{i} = \boldsymbol{s}^{i+1} - \boldsymbol{s}^{i} - \boldsymbol{\beta}_{esp}^{i} \tag{3.5}
$$

$$
\boldsymbol{p}_{vo}^{i} = \boldsymbol{s}^{i+1} - \boldsymbol{s}^{i} - \boldsymbol{\beta}_{vo}^{i} \tag{3.6}
$$

$$
\boldsymbol{p}_{gps}^{i} = \boldsymbol{s}^{u_i} - \boldsymbol{s}_{gps}^{i} \tag{3.7}
$$

$$
\boldsymbol{p}_{anchor}^{ij} = \boldsymbol{s}^{v_{ij}} - \boldsymbol{s}_{init}^{v_{ij}} - \delta \boldsymbol{s}_{anchor}^{i} \tag{3.8}
$$

where $\boldsymbol{C}_{esp}$, $\boldsymbol{C}_{vo}^{i}$, $\boldsymbol{C}_{gps}^{i}$, and $\boldsymbol{C}_{anchor}^{i}$ are respectively error covariances of ESP measurements, visual odometry at the *i*-th frame, the *i*-th GPS measurement depending on number of visible satellites, and the *i*-th anchor constraint. M, N, and K are numbers of frames, GPS measurements, and identified anchors. L_i is the frame length of the *i*-th anchor. u_i is the time for the *i*-th GPS measurement, and v_{ij} is the time of the *j*-th anchor frame in the *i*-th anchor, all in frame unit. $\delta \boldsymbol{s}_{anchor}^{i}$ denotes the *i*-th anchor constraint, i.e. unified optimal adjustment of all frames in the *i*-th anchor. We set the yaw dimension in all $\delta \boldsymbol{s}_{anchor}^{i}$ zero as the 2-axis 1D matching does not count orientation change. We use the version of Levenberg-Marquardt solver implemented in GTSAM [15, 16] to optimize the final objective function.

Final Image Generation

After the optimal states have been estimated, we render all orthographic views onto the initial satellite image background to generate final high resolution

roadway image. To deal with overlapping areas, we first compute weights for every pixel in each frame, in inverse proportion to the pixel's distance to the camera. Only the pixel with the highest weight will be used in the final map. This is because areas closer to the camera have higher resolution and they are not as effected by the changes in roll and pitch angles during the drive. We also find out that keeping only the highest weighted pixel usually yields better image quality than other strategies such as weighted averaging.

3.3.3 Anchor-wise Incremental Optimization

The global optimization framework we just described requires all measurement being collected. However this is not desirable in practical mapping applications, especially when the data sequence is long. Therefore we further design an anchor-wise incremental optimization algorithm that allows map updating to be more efficient, as described in Algorithm 1. We combine our above method with EKF. Anchor-based matching and optimization are only triggered at anchor end frames. In addition, we apply optimization and updating only for the most recent k_{opt} anchors, one fixed anchor is also added to ensure continuity of the final trajectory. In our implementation we set k_{opt} as 3.

3.4 Experimental Results

In this section we present experimental results. We use an Audi A7 as our test platform, its original GPS, ESP, and an extra added forward looking (zero-tilt calibrated) color camera are used for data collection. Frame rates (fps) of the

Algorithm 1 The incremental optimization procedure for map updating

for $t = 1 : M$ **do**
  if $t == 1$ **then**
    Set $\boldsymbol{s}^1$ as $\boldsymbol{s}^1_{gps}$
  else
    if Detect the end of anchor i, i.e. $t == v_{iL_i}$ **then**
      if $i > k_{opt}$ **then**
        Compute anchor constraints $\delta \boldsymbol{s}^{(i-k_{opt}):i}_{anchor}$, set $\delta \boldsymbol{s}^{i-k_{opt}-1}_{anchor}$ zeros
        Optimize and update $\boldsymbol{s}^{v_{(i-k_{opt}-1)1}:v_{iL_i}}$
      else
        Compute anchor constraints $\delta \boldsymbol{s}^{1:i}_{anchor}$
        Optimize and update $\boldsymbol{s}^{v_{11}:v_{iL_i}}$
      end if
    else
      Propagate $\boldsymbol{s}^M$ using EKF with $\boldsymbol{s}^t_{esp}$, $\boldsymbol{s}^t_{vo}$, and $\boldsymbol{s}^{u_{k'}=t}_{gps}$ if has new GPS reading
    end if
  end if
end for

used GPS, ESP, and camera are respectively 1, 50, and 30. We stitch image patches from Google maps as the low resolution satellite image prior, assuming no distortion. The stitched map has resolution of $23.5cm/pixel$. We test our algorithm at Clipper Drive near Belmont, CA. The test sequence has duration of $215s$ and length of $1443m$.

We first evaluate the accuracy of our algorithm. To measure accuracy quantitatively, we select a 2000 frame sub-sequence (first $66.7s$ in attached video, length $439m$) and set one measure point every second. We manually align input orthographic views of measure points with the satellite image, and use them as ground truths. As comparison to our method, we also evaluate 3 other methods using the same data. The compared methods are: (1) Extended Kalman Filtering (EKF): applying EKF on GPS and ESP, without using any image information. (2) Single-Frame Correlation (SFC): using GPS, ESP, visual odometry, and

Table 3.2: Quantitative results of different methods.

	EKF	SFC	CI[6]	**Proposed**
mean error(m)	4.02	2.94	1.16	**0.16**
max error(m)	6.04	12.32	2.74	**0.78**

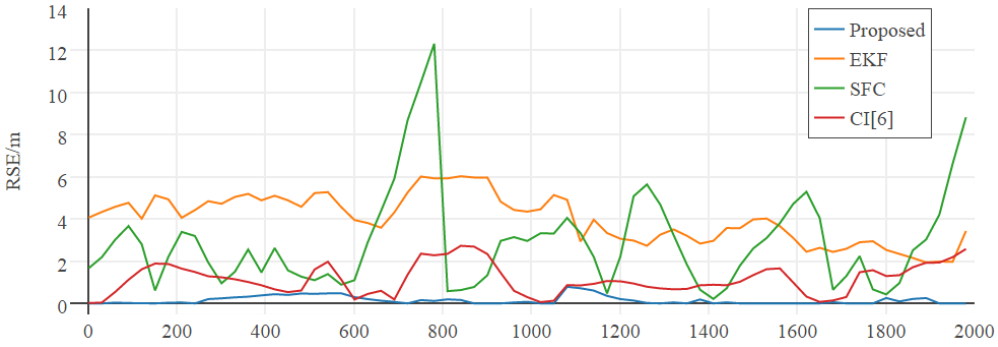


Figure 3.5: Diagram of errors over time.

camera-satellite image matching. Rather than anchor identification and anchor-wise constraints, single frame matching correlations are directly used for final optimization. (3) Calibrated Intersections (CI): this is similar to the method described in [27]. GPS, ESP, and visual odometry are used for positioning. The road network is divided into road segments, with the intersections of segments localized accurately. Table 3.2 and Figure 3.5 show quantitative results of all evaluated methods.

From Table 3.2 and Figure 3.5, it can be seen that our method achieves higher accuracy than all three compared methods. In the EKF method no image information is used, thus high accuracy localization cannot be achieved due to limited precision of the low-cost GPS. In the SFC method, in addition to GPS and ESP, visual odometry and camera-satellite image matching are also used. This improves overall accuracy and produces lower mean error than EKF. However as no anchor strategy is used and all single frame correlations are directly sent to

optimization, the final estimated states are severely affected by frames that are highly ambiguous and inaccurate. This explains why SFC produces high maximum error. In the CI method [27], 3 road segment intersections (around frame 10, frame 810, and frame 1680) are detected. While intersections are localized accurately, the distance between two intersections are relatively long and drifting error propogates without being refined by any precise location constraints. This leads to error accumulation between intersections and impairs the final performance of CI. Our proposed method outperforms CI as it is able to detect anchors adaptively and calibrate itself using detected anchors. Our proposed anchor strategy automatically turns off inaccurate image matching results, thus avoids the problem of SFC. In this sub-sequence 8 anchors are identified.

In Figure 3.6, we show qualitative results of final maps generated by different methods. It can be seen that EKF produces low accuracy and images are not satisfactorily positioned, mainly due to GPS does not have sufficient precision. In SFC sometimes the images are positioned more accurately but sometimes not, also the final map appears to be jagged or twisted. This demonstrated that SFC fails to generate accurate map because inaccurate single frame correlations severely interferes final state estimation. Both CI [27] and the proposed method generate high quality maps. The proposed method has better consistency with the original map as aligned anchors impose stronger consistency constraints than calibrated intersections.

Figure 3.7 shows the final generated high resolution roadway image and Figure 3.8 shows example patches from Figure 3.7. In Figure 3.8, the first column shows two examples where the generated map shows clear lane marks, the second and third columns show examples that the generated map provides new

Figure 3.6: Qualitative results of different methods. Top to fourth row: results produced by EKF, SFC, CI [27], and proposed method. Bottom row: the original satellite image. Areas marked red show that the proposed method has better consistency with the original satellite image.

detailed textures that cannot be observed from the satellite image. These examples imply the usefulness of the generated map in applications such as vehicle localization and intelligent self driving.

Though we are able to create generally satisfactory and useful maps, our method still has a drawback. As shown in the last column in Figure 8 some

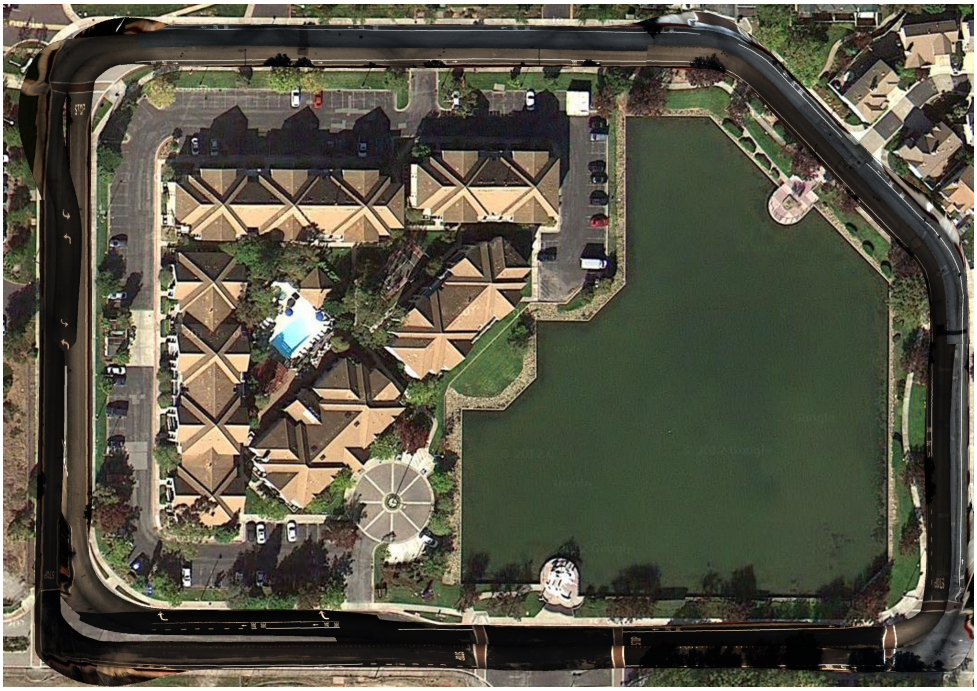


Figure 3.7: The final high resolution road image generated in our study.

Figure 3.8: Examples from the final generated high resolution road image.

parts in the final map are distorted. This is mainly caused by the curvature of the road and/or sudden vehicle speed change. One way to solve this problem is to estimated full 3D camera poses including pitch and roll angles in visual odometry, instead of only 2D poses in Equation 3.3. However we choose not to do so as estimating 3D poses from stereo requires intensive computational

power [62], which would significantly increase the hardware requirements.

The multimedia attachment shows an example of our incremental optimization procedure, as well as detailed anchor positions.

3.5 Conclusions and Future Work

A case study for generating high resolution roadway images that are consistent with an existing satellite image was presented. Our method uses publicly accessible map resources and conventional low-cost sensors, indicating more potential probe vehicles and lower mapping cost. Our generated maps can be easily embedded into existing maps due to its high consistency. An anchor-based method was proposed to improve state estimation accuracy and mapping consistency.

This work can be extended to deal with curvature on the road, compensating for roll and pitch effects, and generating high resolution image that is invariant to lighting condition. Such a map can be used as vision aiding cue in localization methods.

CHAPTER 4

YOU ARE HERE: MIMICKING THE HUMAN THINKING PROCESS IN READING FLOOR-PLANS

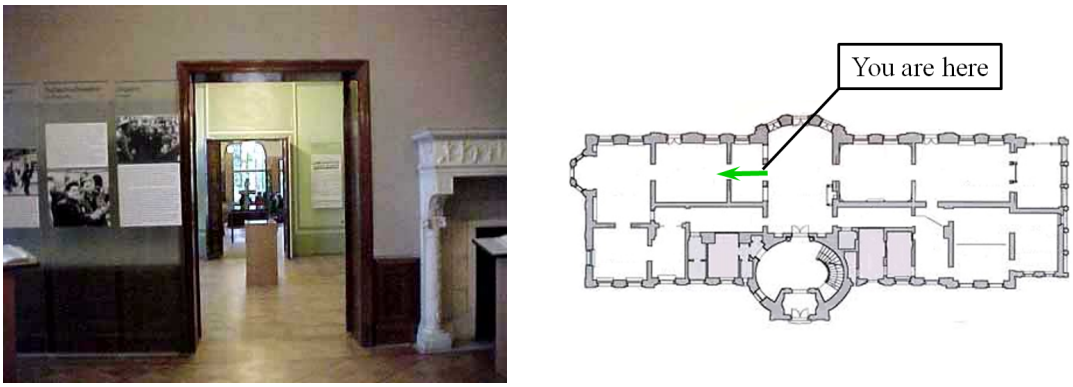
4.1 Introduction

Floor-plans contain useful structure information about building interiors, and they are easy to acquire [63] as all buildings have the plans. Floor-plan related problems, therefore, have received a lot of attention from the computer vision research community [5,7,21,22,38,42].

Floor-plan is also widely used in daily life. It is commonly used as a guide in large buildings, such as museums, malls, and laboratories. There are two common scenarios in using the floor-plan. In the first scenario, the floor-plan is found at a fixed location. There is often a "You are here" arrow in the plan to help a viewer quickly find out where he or she is located. In the second scenario, a tourist has the floor-plan in hand. In this case, there is no the "You are here" arrow in the plan. Instead, the tourist needs to find out where he or she is by walking around, comparing the observed scenery with the floor-plan, and eventually figuring out the correct location.

Our goal is to help human users by solving the localization problem fully automatically. More specifically, our system stores a floor-plan as prior, takes a video stream of what a tourist sees, and estimates the tourist's current position and orientation in the floor-plan. Figure 1 shows an example of the input and output of our system.

Several reasons make our goal challenging to accomplish. Firstly, the floor-



(a) (b)

Figure 4.1: Our system takes a video stream and a floor-plan, and outputs the position and orientation of the current frame in the floor-plan.

plan does not provide any information about color or texture. Thus, it is hard to use methods based on image feature matching, which rely on previously stored features. Secondly, buildings are composed of repetitive structures. Those repetitive structures, such as corner and corridor, can be observed at multiple locations. Thirdly, the floor-plan can be inconsistent with the real-world. It outlines the building structure, but does not include the furniture. Thus, it is difficult to apply existing conventional techniques in our task:

SLAM Vision-based methods for Simultaneous Localization and Mapping (SLAM) [4,9,14,18,33,47,57] are able to create representational feature maps, and estimate the observer's motion. Although in some applications floor-plans are used, an annotated feature map is still needed to perform localization. Thus, it is difficult to use those methods to localize in a building that has not been mapped beforehand by similar algorithms. SLAM methods are specialized in effective camera tracking and map creation, rather than finding associations between the observed structure and the floor-plan.

Image Retrieval Image retrieval methods localize the camera by matching

image features with a database of images with known positions [29, 40, 70], or a prior 3D reconstruction of the environment [28, 35]. However, in our case we do not have such rich prior information. Only a floor-plan is known when the system starts, which makes our problem essentially different from the image retrieval localization problem.

Vehicle Localization Vision-based vehicle localization methods [6, 26, 66] find the vehicle's current location by analyzing the video stream captured on the vehicle and the topology of the road network. It is difficult to directly transfer these methods to apply to our task, because in the indoor environment the camera moves freely in a 3D space, while vehicle cameras moves constrainedly on the lane.

We solve the problem by gaining inspirations from the human thinking process: a very common localization approach a human would use is to observe one room's size and structure, compare the observation with the floor-plan, and come up with a few possible options. Then move to another room and do the same process, until being certain of the position. Our algorithm mimics this human thinking process. In other words, we help a human user solve the localization problem in the human way.

In this paper we propose: 1. A novel and useful task of indoor localization with only a camera and a floor-plan, without any other prior information. Thus it can be used in any building that has a floor-plan and does not require feature map reconstruction as the preparation. 2. An efficient algorithm that localizes the camera in a floor-plan by mimicking the human thinking process.

4.2 Related Work

In this section, we examine related previous work in two categories: computer vision research that uses floor-plans, and indoor localization techniques.

Floor-plans in Computer Vision

The floor-plan data has been catching more and more attention in the computer vision research community. It is used in several recent works. In Martin-Brualla et al. [5], floor-plans are used to solve the 3D jigsaw puzzle, which is to find the correct layout of a set of disconnected pieces of 3D reconstruction. In Cabral et al. [7] and Furukawa et al. [22], the floor-plan structure is reconstructed from a set of indoor images. In Liu et al. [42], floor-plan priors are used to accurately register image textures onto walls. In [21,52,74,77] computer vision techniques are applied for indoor structure estimation and indoor scene understanding from images or videos. Though those works are inspirational and useful in their own application scenarios, they can not be directly applied to solve our problem. The work most similar to ours is [38], where an omnidirectional camera is used for localization in a floor-plan. Besides using an unconventional camera, their method is based on overlapping frame detection, which requires closed loops in the camera motion trajectory. Thus, it is difficult to use their method in our task.

Indoor Localization

Indoor localization has been an active area of research in the mobile computing community. Various indoor localization approaches have been proposed, such as indoor localization based on signal beacons [64], magnetometer [12], and sound/ultrasound [68, 69]. Although these methods show promising results, they suffer from an inevitable problem of requiring specialized infrastructures or sensors, which makes them difficult to scale up.

Another line of work makes use of widely existing wireless signals such as GSM and WiFi [32, 55], or the ubiquitous geomagnetism [76]. Methods of this type show better scalability as they do not require additional dedicated beacons or sensors, and have been made publicly available by mobile software such as Google Maps Indoor. However, these methods are not free from mapping survey stages, i.e. signal fingerprint maps are needed for localization. This creates a bottleneck for scalability as well as maintenance difficulties.

The approach that is the closest to ours uses only self-motion estimation and floor-plans [37, 39, 54]. This approach shows strong scalability as neither special sensor nor mapping stage is needed. Also, maintenance of this approach is easy because the building structure seldom changes. The problem of this approach is that a long distance is needed for localization to converge—imagine finding the way with eyes closed. Thus, we propose to improve this approach by giving the system the ability to see. We show in the experiments that visual information significantly improves the localization performance, and our method outperforms the theoretical upper-bound of motion-only methods.

Beyond the mobile computing community, indoor localization has also been

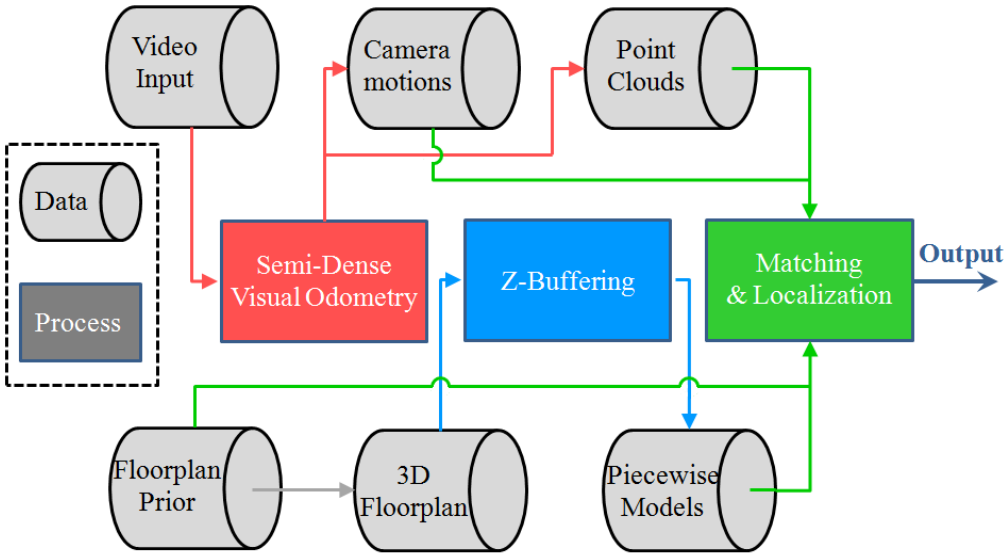


Figure 4.2: An overview of our system.

studied in [31, 41, 43, 80]. Those approaches use vision sensors along with various other types of sensors, such as WiFi receiver, depth sensor, inertial sensors, and LIDAR. Although those methods provide reliable and effective solutions for indoor localization, it is expensive for them to scale to large number of buildings and large number of users. The problem of self-localization with only a camera and without any other prior database but a floor-plan, while being attractive and potentially useful, remains unstudied.

4.3 System Pipeline

This section describes the overall pipeline of our system, as illustrated in Figure 2. Our system consists of three major steps.

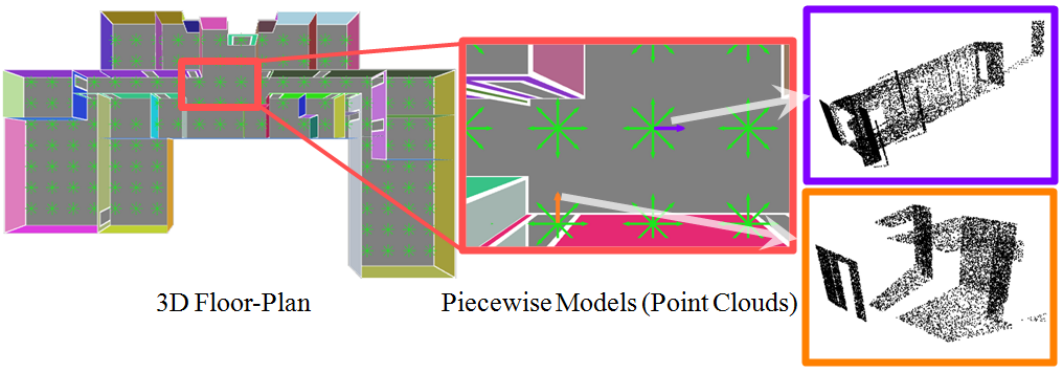


Figure 4.3: An example of generating the piecewise models.

Motion Estimation and Reconstruction

We use the publicly available software of Semi-Dense Visual Odometry (SDVO) [18, 19] to simultaneously estimate the camera motion and reconstruct the 3D structure from a video taken by the camera. Comparing to other existing methods such as the well known structure from motion (SfM) [34, 61, 73] and multi-view stereo (MVS) [23], SDVO has the advantage of producing rich information of the building structure as well as being computationally inexpensive. In contrast, the reconstruction of SfM is too sparse to conduct effective analysis with, and the reconstruction of MVS consumes significant amount of computational power, which makes it not applicable for real-time computation.

Generating Piecewise Floorplan Models

We generate the 3D floor-plan model by lifting up the 2D floor-plan. We set multiple unique viewpoints in the 3D floor-plan, then for each viewpoint, we obtain a piecewise floor-plan model by setting a virtual camera at the viewpoint, and applying the z-buffering technique in computer graphics to preserve

only the visible part. The z-buffering procedure produces a depth image. Given a depth map and the camera intrinsic, one can construct a point cloud where one point corresponds to one pixel in the depth image. However, such point cloud is spatially non-uniform, i.e. nearby objects have more pixels, thus have higher point density than faraway objects. To overcome this problem and to uniformly represent the piecewise floor-plan, we sample each pixel with a probability proportional to the area of its back-projected square region. More precisely, $p_i \propto depth_i^2$, where p_i is the sampling probability of pixel i. Figure 3 shows an example of piecewise model generation.

Matching and Localization

New video frames are used for an online process of matching and localization. In the beginning, the algorithm does not have an accurate answer. Every position in the indoor space has the same possibility. As the camera moves, 3D models are reconstructed every two seconds. Then matching is performed using the reconstructed model, which helps the estimation converge to the correct current camera position. Matching and localization will be described in Section 4 and 5.

4.4 Matching by Mimicking Human Logic

When reading the floor-plan, humans often focus on the overall shape of the room, the structure of walls, and the room's size and space. Our matching algorithm tries to mimic this human thinking process, and it consists of three steps:

full point cloud matching, reliable structural line matching, and conservative free space matching.

Full Point Cloud Matching

Full point cloud matching matches the point cloud reconstructed from camera $M \in \mathbb{R}^{3\times m}$ and each piecewise floor-plan model $N^{(i)} \in \mathbb{R}^{3\times n_i}$, where m and n_i being numbers of points. We perform full point cloud matching for two reasons. First, we want to search for piecewise models that match well with the reconstruction. However, there can be a slight offset even for the well-matched piecewise models because the actual camera can have an arbitrary pose, in contrast to the piecewise models that are captured with fixed camera viewpoints. Thus we perform alignment between the two point clouds, i.e. it finds $R \in \mathbb{R}^{3\times 3}$ and $t \in \mathbb{R}^{3\times 1}$ such that $M' = RM + t$ is aligned with $N^{(i)}$. Second, we want to compute similarity between the aligned point clouds by $S^{(i)}_{FULL}(M', N^{(i)})$, which is used to help localization.

We use the well known iterative closest point (ICP) method [78] to align the two point clouds. We constrain that the alignment is valid only if it is within a viewpoint grid of $N^{(i)}$. In other words, M can only be translated or rotated slightly to be alignable with $N^{(i)}$, otherwise M is too far away from $N^{(i)}$ to be considered for matching, thus similarity is zero. Moreover, the number of piecewise models is relatively large, and for most of them, $N^{(i)}$ and M are fairly far away. Thus, we only perform ICP for M and a selected subset of piecewise models to avoid unnecessary computation. A piecewise model is selected with M using ICP, only if it satisfies

$$\left(\sum_{k=1}^{m} \mathbf{1}\{||M_k - f_n(N^{(i)}, M_k)|| < d_k\}\right) \geq \alpha m \tag{4.1}$$

where $\alpha \in (0, 1)$, $f_n(N^{(i)}, M_k)$ returns the nearest neighbor of M_k in $N^{(i)}$ using k-d tree, and d_k is the maximum motion with within-grid rotation and translation, which is defined as:

$$d_k = \beta||M_k|| + \gamma \geq \max_{R,t} ||RM_k + t - M_k|| \tag{4.2}$$

The selection process is controlled by three constants α, β and γ.

For a piecewise model $N^{(i)}$, if (1) is satisfied, we apply ICP to align M with it. Then we measure the similarity between two aligned point clouds as

$$S^{(i)}_{FULL}(M', N^{(i)}) = \frac{1}{m} \sum_{j} \frac{\lambda_1}{\lambda_j} I_j \tag{4.3}$$

where $\lambda_{j+1} > \lambda_j > ... > \lambda_1 > \lambda_0 = 0$ are constants of distance thresholds, and I_j counts inliers between λ_j and λ_{j+1}

$$I_j = \sum_{k=1}^{m} \mathbf{1}\{\lambda_j \leq ||M'_k - f_n(N^{(i)}, M'_k)|| < \lambda_{j+1}\} \tag{4.4}$$

Reliable Structural Line Matching

The SDVO reconstruction is based on image edges, thus the full point cloud contains outlines of any objects, such as chairs and desks. This causes troubles

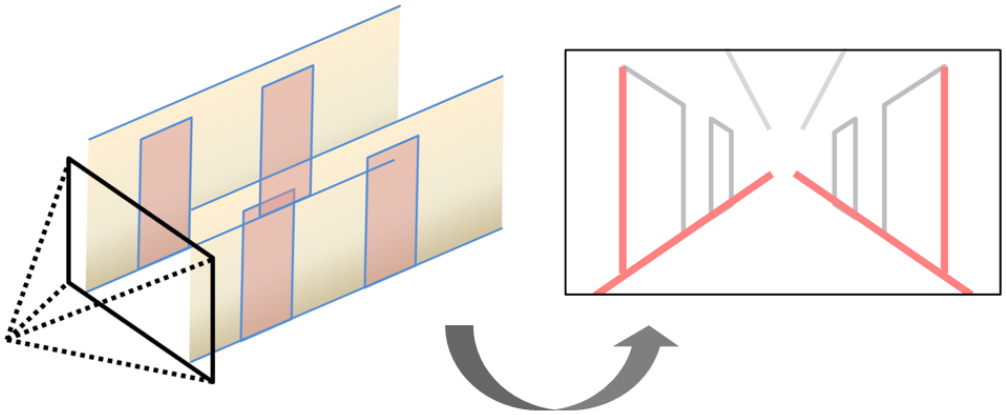


Figure 4.4: Toy example of reliable structural lines (red).

for accurate matching because the floor-plan only describes the empty building and does not have any information about outlines of any objects. To tackle this problem, we perform reliable structural line matching: matching using only long straight lines in the SDVO reconstruction, which often correspond to building corners or doorframes, and lines in the piecewise floor-plan models.

Reliable structural lines are defined as long line segments in the depth image. We find the reliable structural lines by first detecting a set of line segments in the depth image. Then we greedily find line segment pairs that are near to each other with a similar angle, and merge them to get longer line segments. Note that we detect reliable structural lines using the 2D depth image rather than the 3D point cloud. This is not only because line detection is often more effective in 2D than 3D, but also because lines that are nearer to the camera are more likely to be selected. Nearer lines often have larger displacement between images, and their depth estimations tend to be more accurate and reliable. Figure 4 shows an example of reliable structural line detection.

After reliable structural lines are extracted, we represent them as a point cloud, and use the similar method described through Equation (1)-(4) to deter-

mine its similarity to the piecewise models, i.e. $S^{(i)}_{RSL}(M'_{RSL}, N^{(i)}_{RSL})$, where M'_{RSL} is the aligned point cloud of reliable structural lines (if alignable) and $N^{(i)}_{RSL}$ is the point cloud of edges of a piecewise model.

Conservative Free Space Matching

Free space is the space from the camera to the nearest obstacle. Because indoor objects are not included in the floor-plan, the free space observed by the camera will not always match exactly with the free space measured using the floor-plan. However, the observed free space should always be a subset of the free space measured at the correct position in the floor-plan. This property can be used for faster and more accurate localization.

Conservative free space matching takes a reconstructed point cloud and an indoor camera pose as inputs. For the point cloud, we set several directions and estimate the distance to the nearest object in each direction. In the *i*-th direction, we first compute a discrete 1D signal C^i, where $C^i(d)$ equals to the number of pixels that has depth d. If one pixel has a depth value of d, its depth is computed by its inverse depth $1/d$, which is proportional to the between-frame pixel displacement. Assuming the error in pixel displacement is Gaussian, the error in the inverse depth estimation is also Gaussian. Thus, the probabilistic depth distribution is computed as

$$P^i(d) = \sum_{d'} C^i(d') G(\frac{1}{d'} - \frac{1}{d}) \tag{4.5}$$

where $G(x)$ characterizes the Gaussian error in inverse depth estimation

$\mathcal{N}(0, \sigma_{id})$. Once we have $P^i(d)$, we compute the free space span as

$$U^i = \begin{cases} \underset{d<D_{max}}{\arg\min}\, P^i(d) \geq Q^i(d) & \text{if such } d \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where $Q^i(d) = \eta/d$ is the threshold. The free space estimation is conservative as we set U^i as zero whenever all $P^i(d)$ is below the threshold. This is due to the existence of texture-less objects, such as a blank wall. Texture-less objects are not reconstructed in a monocular vision system, which leaves areas with unknown depth. If no sufficiently dense obstacle is detected in a certain direction, we assign zero free space in that direction to minimize the false positive. An example of finding the free space span is shown in Figure 5.

Similarly to $U = \{U^i\}$, the free space span of a camera pose in the 3D floor-plan $V = \{V^i\}$ is computed with a very small value of σ_{id}. We compute the final similarity of conservative free space matching by measuring how well the subset rule is obeyed:

$$S_{FS}(U, V) = \prod_i \exp(-\delta \cdot \mathbf{1}\{U^i > V^i\}) \tag{4.7}$$

4.5 Localization

Localization is performed by applying particle filter using the floor-plan, camera motions, and similarities computed from the three types of matching described above. To begin with, we first briefly summarize the properties of the

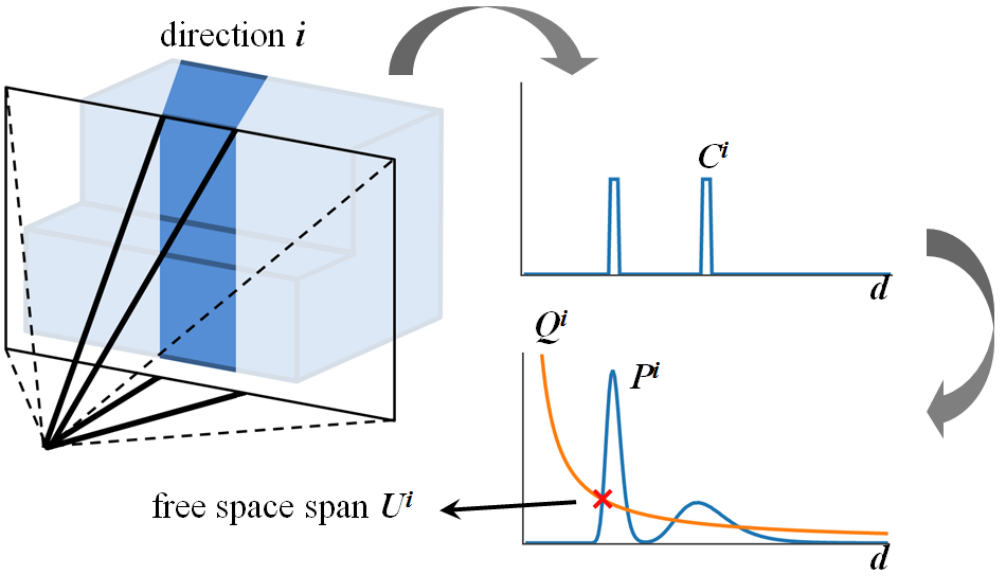


Figure 4.5: An example of finding the free space span.

three types of matching similarities, as listed in Table 1. When the value of S_{FULL} or S_{RSL} is high, the possibility that the matched piecewise model being the correct one is medium, as the structure of full point cloud and reliable structural line is often distinctive but not unique. As S_{FS} measures the inclusion of free space, it has limited selectivity when the value is high, thus low confidence. When S_{FULL} is low, the matched piecewise model can still be the correct one due to the existence of not-mapped objects. It is similar when S_{RSL} is low, but the chance of matching with the correct piecewise model is smaller as the outline of a common object is less likely to become a reliable structural line. When S_{FS} is low, the rule of free space inclusion is violated. Thus, the matched camera pose is very likely to be incorrect as our free space estimation is conservative. Table 1 summarizes by listing the effectiveness of distinguishing the correct location in the first column and the effectiveness of excluding incorrect locations in the second column.

We formulate localization as a particle filtering process, where each particle

Table 4.1: Comparisons between different cues from experiments: high similarity values promote correct locations, and low similarity values eliminate incorrect locations.

	high val. conf.	*low val. conf.*
S_{FULL}	22.4% (*medium*)	7.9% (*low*)
S_{RSL}	17.7% (*medium*)	14.5% (*medium*)
S_{FS}	6.8% (*low*)	89.2% (*very high*)

(a) (b)

Figure 4.6: An example of the ground-truth labeling process: first roughly estimate the camera pose, and overlay its view in the 3D floor-plan with the actual image, as shown in (a). Then change the 6-DoF camera pose with an interactive interface, until the two views are consistent as shown in (b).

represents a 6-DoF hypothesis of the current camera pose. Initially, the particles are distributed uniformly in the indoor space. As the camera starts moving, camera motions are estimated and new 3D point clouds are reconstructed and matched every several frames. We apply the estimated camera motion to the particles, and adjust the possibilities of particles using the reasoning presented in Table 1. More formally, the algorithm takes a video stream as input, the floor-plan as prior, and estimates the camera pose of each input frame. Algorithm 1 outlines the localization process.

Algorithm 2 The online-localization process

Input: video stream
Prior: floorplan
Output: camera poses of frames

1: Generate piecewise models
2: Initialize random particles with equal weights
3: Take in a new frame, update reconstruction M_j, estimate motion
4: Apply motion to particles
5: **if** M_j is complete **then**
6: Compare M_j with all piecewise models, compute $\{S^{(i)}_{FULL}\}$ and $\{S^{(i)}_{RSL}\}$
7: Compute M_j's free space U_j
8: **for** each particle **do**
9: Compute floorplan free space V, compute S_{FS}
10: Adjust weight according to S_{FS}, the spatially nearest $\{S^{(i)}_{FULL}\}$ and $\{S^{(i)}_{RSL}\}$
11: **end for**
12: Kill particles with low weights
13: Replace dead particles by weight-based resampling
14: Begin reconstructing M_{j+1}
15: **end if**
16: Compute maximum-likelihood camera pose estimation with current particle distribution
17: Go back to 3

4.6 Dataset

There are many existing datasets for indoor location [30, 53]. However, most of them do not apply to our scenario for two reasons. Firstly, they are captured by a fixed-height platform and do not allow fully free 3D camera motion. Secondly, not all datasets provide ground-truth location labels in reference to the floor-plan. Thus, we created our own dataset to test our algorithm. We use a hand-held cellphone camera to capture videos when walking through different rooms. We allow free 3D camera motion during the capturing. Our dataset consists of six videos sequences, including $21,700$ image frames, with total walking distance of $205m$. We manually created 586 6-DoF camera location labels as the

ground-truth. Figure 6 describes the location labeling process.

4.7 Experiments

We ran our experiments on a PC with a Intel-i7 processor. The algorithm was implemented in serial C++ code. We tested our algorithm in a building of size $875m^2$. The building floor-plan only marks walls and doors. We created the 3D floor-plan using room height of $3m$ and door height of $2.3m$. We set a $2m$ grid in the building, and set 12 piecewise model viewpoints with different orientation at each grid. 1608 piecewise models were generated within 10 minutes of computation. The total file size of all generated piecewise models was $74.3MB$.

To evaluate the performance of algorithms, three types of evaluation metrics were used: *Succeed Distance* measures the total walking distance from the beginning of the video to localization success. We define localization to be success if, for all afterwards frames, the estimated position and orientation remain within $1.5m$ and 20° to the ground-truth label. *Position Accuracy* and *Orientation Accuracy* measure the average position and orientation error after localization success, respectively in meters and degrees. In all three metrics, the primary goal of our task is to minimize *Succeed Distance*. The shorter it is, the faster the correct location can be found, and the less users need to walk. *Position Accuracy* and *Orientation Accuracy* are secondary compared to *Succeed Distance*, because the accuracies are always below $1.5m$ and 20°, and it is more important to find out the correct overall location than to improve the precision by several centimeters or degrees.

We applied our algorithm to all six videos in our dataset. In Figure 7-9 we

Table 4.2: *Succeed Distance* (smaller the better), *Position Accuracy*, and *Orientation Accuracy* of the Mobile Computing Theoretically Best (MCTB) method, the Scan-Matching Particle Filter (SMPF) method, and the proposed method, on a public TUMindoor (TUMI) dataset and our dataset.

	Succ. Dis.		*Pos.*		*Orien.*	
dataset	TUMI [30]	Ours	TUMI [30]	Ours	TUMI [30]	Ours
MCTB [37,39,54]	$51.92m$	$26.73m$	$0.65m$	$0.82m$	$1.63°$	$5.52°$
SMPF [9]	$16.62m$	$25.77m$	$\mathbf{0.51m}$	$0.72m$	$1.40°$	$\mathbf{5.21°}$
Proposed	$\mathbf{15.77m}$	$\mathbf{19.02m}$	$0.53m$	$\mathbf{0.50m}$	$\mathbf{1.29°}$	$5.39°$

Table 4.3: *fps* of S_{FULL}, S_{RSL}, S_{FS}, localization and total algorithm, as well as memory consumption and *Success Distance* of our full method and its real-time, 33% Piecewise Model version (purple curve in Figure 11). Input *fps* is 29.

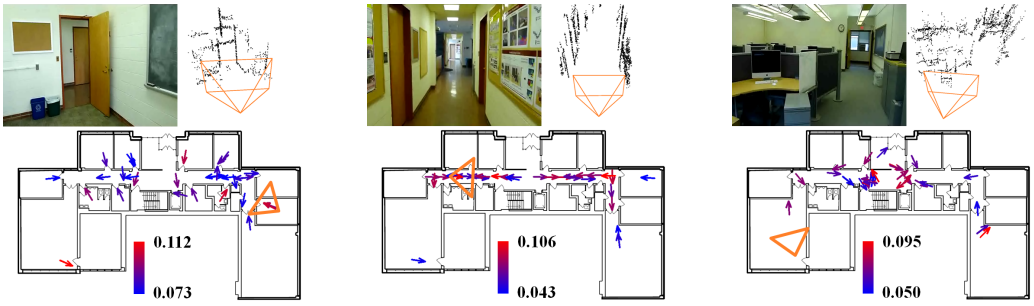
	S_{FULL}	S_{RSL}	S_{FS}	*loc.*	*total*	*memory*	*Succ. Dis.*
full	16	148	$> 1k$	355	14	$83.0M$	$\mathbf{19.02m}$
33%PM	42	266	$> 1k$	394	**33**	$\mathbf{26.5M}$	$19.39m$

show examples of the three proposed matching methods. It can be seen that all matching methods provide useful information of the camera location. Full point cloud matching and reliable structural line matching produced both good and bad results. Bad matching occurred due to the ambiguity of room structure information, and the inconsistency between the floor-plan and the real world, as we discussed in the introduction. Despite of the possibility of producing bad matching results, the two proposed matching methods improve the final localization performance when all matching results of the whole video are considered, as shown in Table 2. The conservative free space matching is affected by indoor objects. It provides a good way of excluding incorrect hypotheses.

We compared our method with two baseline methods. First is the Mobile Computing Theoretically Best (MCTB) method [37, 39, 54]. [37, 39, 54] propose various approaches to estimate the motion trajectory more accurately, with an inertial sensor and/or a camera. The estimated motion is then used for localiza-

tion with a particle filtering process similar to ours. Thus, the theoretically best performance of them is achieved when the motion estimation has zero-error. We directly used motions obtained from the ground-truth location labels for the MCTB method. Second is the Scan Matching Particle Filter (SMPF) method [9], where we obtained 2D scans from the estimated depth images and used [9] to match scans with the floor-plan. Beside our dataset, we also evaluated on a public dataset of TUMindoor [30]. We used five video sequences in the Ground II level, with total walking distance of $606m$. For each video, we ran all algorithms five times because of the random particle initialization. Table 2 lists the results. It is noticeable that SMPF and our method outperform MCTB on both datasets. For TUMindoor dataset, our method shows a similar performance to SMPF. However, our method outperforms SMPF when applied to our dataset, especially in the primary goal of *Success Distance*. TUMindoor dataset contains no objects or furniture, but only building structures. Our dataset, on the other hand, contains more practical scenarios with many objects and furniture. This evaluation demonstrates our method's robustness and effectiveness to more complex and realistic environments. For the primary goal of *Succeed Distance*, our method achieves 28.8% and 26.2% improvement compared to MCTB and SMPF, respectively. Figure 10 shows the average errors in position and orientation against the walking distance. The compared methods show high error at the end because they fail to find the correct location in two videos. It can be seen that our algorithm decreases the errors faster, which means the hypotheses converge to the correct solution more effectively. Please refer to the video attachment for more detailed results.

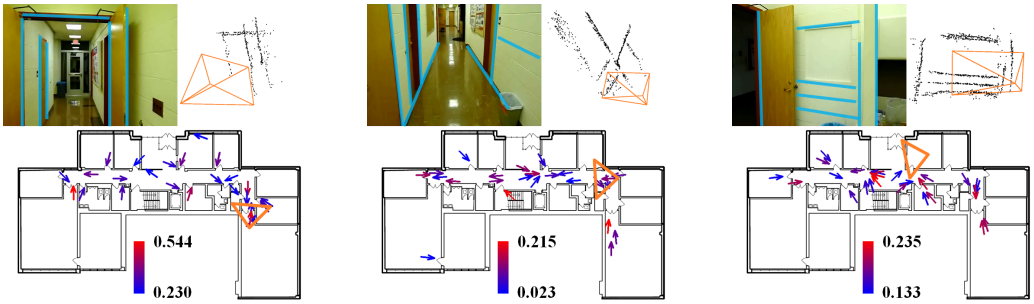
Our method significantly outperforms [37, 39, 54], but it also requires more computation due to performing more matching. In the second experiment, we



(a) (b) (c)

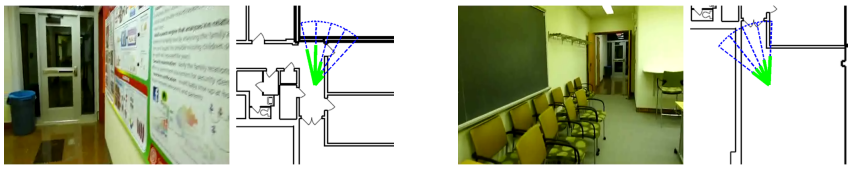
Figure 4.7: Input image, reconstructed point cloud, and full point cloud matching results. Arrows show 30 matches with highest similarity S_{FULL}, color corresponds to similarity. Orange triangle shows the ground-truth camera pose. (a) and (b) show good matching results, (c) shows a failure case because no arrows are near the ground-truth.

evaluate the efficiency of our method. Table 3 lists the frame rate of our matching and localization procedures. The video frame rate of our data is 29. It can be seen that the bottleneck is the computation of full point cloud matching S_{FULL}. To achieve real-time localization, we applied two strategies to improve the frame rate of S_{FULL}: 1. use less reconstructions and compute S_{FULL} less frequently, i.e. compute S_{FULL} only once or twice in every three reconstructions; 2. use less piecewise models, i.e. cut the number of piecewise models by 33% and 66%. Figure 11 shows the experiment results. It can be seen that using less piecewise models is the better strategy, the performance only degrades slightly even using only 33% of all piecewise models. This is because we sampled piecewise models every $30°$, and using 33% of them keeps piecewise models with $0°$, $90°$, $180°$, and $270°$ orientations. These piecewise models obey the manhattan-like structure of the building and capture the most useful structural information of the building. Despite slight degradations, all results in Figure 11 are better than the theoretically best performance of [37, 39, 54]. It also should be noted that linear speed-up can be easily achieved by parallelization.



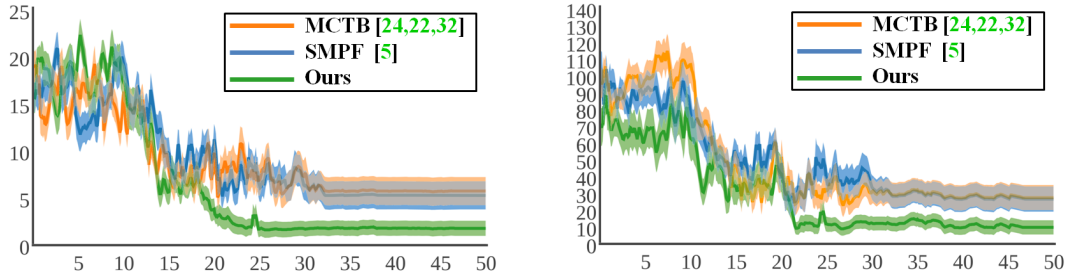
(a) (b) (c)

Figure 4.8: Input image and detected reliable structural lines, point cloud of lines, and matching results. Arrows show 30 matches with highest similarity S_{RSL}, color corresponds to similarity. Orange triangle shows the ground-truth camera pose. (a) and (b) show good matching results, (c) shows a failure case case because no arrows are near the ground-truth.



(a) (b)

Figure 4.9: Detected free space plotted at the ground-truth location. In (a) most free space is detected, in (b) only a subset of free space is detected due to the existence of not-mapped objects.



(a) (b)

Figure 4.10: Errors (y-axis) against walking distance (x-axis in m), our dataset. (a) position error (m), (b) orientation error (°). Area shows 20% standard deviation.

4.8 Future Work

Our future work will be focused in two aspects. First as shown in Figure 11, in the localization process some piecewise models are more useful than the others, which evokes the problem of effective piecewise model selection. Second, rgb-

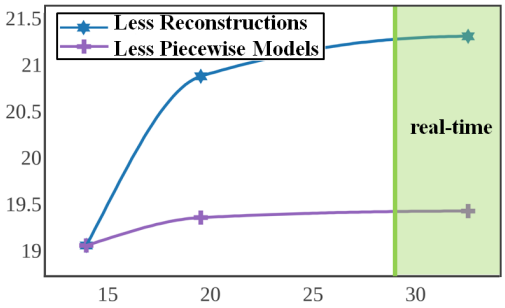


Figure 4.11: *Succeed Distance* (y-axis in m) and *fps* (x-axis) of speedup strategies.

based 3D reconstruction is scale-free, which evokes the problem of automatic scale calibration.

4.9 Conclusion

We introduced a new and useful task of localizing in the floor-plan by using only a camera without any other prior information. We proposed a novel algorithm that is inspired by human logic. We demonstrated the effectiveness and efficiency of our method through experiments.

BIBLIOGRAPHY

[1] Georges Baatz, Kevin Köser, David Chen, Radek Grzeszczuk, and Marc Pollefeys. Handling urban location recognition as a 2d homothetic problem. In *ECCV*, 2010.

[2] Lionel Baboud, Martin Cadík, Elmar Eisemann, and H-P Seidel. Automatic photo-to-terrain alignment for the annotation of mountain pictures. In *CVPR*, 2011.

[3] V Blervaque. Prevent maps & adas d12. 1 final report, ertico–its europe, 2008.

[4] Michael Carsten Bosse. *ATLAS: a framework for large scale automated mapping and localization*. PhD thesis, MIT, 2004.

[5] Ricardo Brualla, Yanling He, Bryan Russell, and Steven Seitz. The 3d jigsaw puzzle: mapping large indoor spaces. In *ECCV*, 2014.

[6] Marcus Brubaker, Andreas Geiger, and Raquel Urtasun. Lost! leveraging the crowd for visual self-localization. In *CVPR*, 2013.

[7] Ricardo Cabral and Yasutaka Furukawa. Piecewise planar and compact floorplan reconstruction from images. In *CVPR*, 2014.

[8] Bruno Caprile and Vincent Torre. Using vanishing points for camera calibration. *IJCV*, 4(2):127–139, 1990.

[9] Andrea Censi. An accurate closed-form estimate of ICP's covariance. In *ICRA*, 2007.

[10] Tat-Jen Cham, Arridhana Ciptadi, Wei-Chian Tan, Minh-Tri Pham, and Liang-Tien Chia. Estimating camera pose from a single urban ground-view omnidirectional image and a 2d building outline map. In *CVPR*, 2010.

[11] David M Chen, Georges Baatz, K Koser, Sam S Tsai, Ramakrishna Vedantham, Timo Pylvanainen, Kimmo Roimela, Xin Chen, Jeff Bach, Marc Pollefeys, et al. City-scale landmark identification on mobile devices. In *CVPR*, 2011.

[12] Jaewoo Chung et al. Indoor location sensing using geo-magnetism. In *Mobile Sys. App. and Srv.*, 2011.

[13] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *ICCV*, 1999.

[14] Andrew J Davison, Ian D Reid, et al. MonoSLAM: real-time single camera slam. *TPAMI*, 29(6):1052–1067, 2007.

[15] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. 2012.

[16] Frank Dellaert and Michael Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[17] Piotr Dollár and C Lawrence Zitnick. Structured forests for fast edge detection. In *ICCV*, 2013.

[18] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular slam. In *ECCV*, 2014.

[19] Jakob Engel, Jürgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *ICCV*, 2013.

[20] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[21] Axel Furlan, Stephen Miller, Domenico G Sorrenti, Li Fei-Fei, and Silvio Savarese. Free your camera: 3d indoor scene understanding from arbitrary camera motion. In *BMVC*, 2013.

[22] Yasutaka Furukawa, Brian Curless, Steven Seitz, and Richard Szeliski. Recon. building interiors from images. In *ICCV*, 2009.

[23] Yasutaka Furukawa and Jean Ponce. PMVS.

[24] Andrew C Gallagher. Using vanishing points to correct camera rotation in images. In *CRV*, 2005.

[25] Andreas Geiger. Monocular road mosaicing for urban environments. In *Intelligent Vehicles Symposium (IV)*, 2009.

[26] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

[27] Chunzhao Guo, Jun-ichi Meguro, Yoshiko Kojima, and Takashi Naito. Automatic lane-level map generation for advanced driver assistance systems using low-cost sensors. In *ICRA*, 2014.

[28] Abner Guzman-Rivera et al. Multi-output learning for camera relocalization. In *CVPR*, 2014.

[29] James Hays and Alexei A Efros. Im2gps: estimating geographic information from a single image. In *CVPR*, 2008.

[30] Robert Huitl et al. TUMindoor: An extensive image and point cloud dataset for visual indoor localization and mapping. In *ICIP*, 2012.

[31] Seigo Ito et al. W-RGB-D: Floor-plan-based indoor global localization using a depth camera and wifi. In *ICRA*, 2014.

[32] Yifei Jiang et al. Ariel: Automatic wi-fi based room fingerprinting for indoor localization. In *Ubiquitous Computing*, 2012.

[33] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *IJRR*, 31(2):216–235, 2012.

[34] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, 2007.

[35] Till Kroeger and Luc Van Gool. Video registration to sfm models. In *ECCV*, 2014.

[36] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Alexander Kleiner, Giorgio Grisetti, and Wolfram Burgard. Large scale graph-based slam using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, 2011.

[37] Kun-Chan Lan and Wen-Yuah Shih. Using smart-phones and floor plans for indoor location tracking. *IEEE Trans. on Human-Machine Systems*, 44(2):211–221, 2014.

[38] Anat Levin and Richard Szeliski. Visual odometry and map correlation. In *CVPR*, 2004.

[39] Fan Li, Chunshui Zhao, et al. A reliable and accurate indoor localization method using phone inertial sensors. In *Ubiquitous Computing*, 2012.

[40] Yunpeng Li, Noah Snavely, Dan Huttenlocher, and Pascal Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012.

[41] Jason Zhi Liang et al. Image-based positioning of mobile devices in indoor environments. *Multimodal Location Estimation of Videos and Images*, pages 85–99, 2015.

[42] Chenxi Liu, Alex Schwing, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Rent3D: Floor-plan priors for monocular layout estimation. In *CVPR*, 2015.

[43] Timothy Liu et al. Indoor localization and visualization using a human-operated backpack system. In *Indoor Pos. and Indoor Nav.*, 2010.

[44] Steven Lovegrove, Andrew J Davison, and Javier Ibanez-Guzmán. Accurate visual odometry from a rear parking camera. In *Intelligent Vehicles Symposium (IV)*, 2011.

[45] Norman Mattern, Robin Schubert, and Gerd Wanielik. High-accurate vehicle localization using digital maps and coherency images. In *Intelligent Vehicles Symposium (IV)*, 2010.

[46] Jun-ichi Meguro, Hiroyuki Ishida, Kiyosumi Kidono, and Yoshiko Kojima. Road ortho-image generation based on accurate vehicle trajectory estimation by gps doppler. In *Intelligent Vehicles Symposium (IV)*, 2012.

[47] Christopher Mei, Gabe Sibley, Mark Cummins, Paul Newman, and Ian Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *IJCV*, 94(2):198–214, 2011.

[48] Ashley Napier and Paul Newman. Generation and exploitation of local orthographic imagery for road vehicle localisation. In *Intelligent Vehicles Symposium (IV)*, 2012.

[49] Minwoo Park, Jiebo Luo, Robert T Collins, and Yanxi Liu. Beyond gps: determining the camera viewing direction of a geotagged image. In *ACM MM*, 2010.

[50] Martin Peter Parsley and Simon Justin Julier. Towards the exploitation of prior information in slam. In *IROS*, 2010.

[51] Oliver Pink and Christoph Stiller. Automated map generation from aerial images for precise vehicle localization. In *Intelligent Transportation Systems (ITSC)*, 2010.

[52] Giovanni Pintore and Enrico Gobbetti. Effective mobile mapping of multi-room indoor structures. *The Visual Computer*, 30(6-8):707–716, 2014.

[53] Poly. Univ. of Catalonia. Barcelona robot lab dataset.

[54] Jiuchao Qian et al. Optical flow based step length estimation for indoor pedestrian navigation on a smartphone. In *Pos. Loc. and Nav. Symposium*, 2014.

[55] Anshul Rai et al. Zee: zero-effort crowdsourcing for indoor localization. In *Mobile Computing and Networking*, 2012.

[56] Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, and Matthew Brand. Skyline2gps: Localization in urban canyons using omni-skylines. In *IROS*, 2010.

[57] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. Slam++: Simultaneous localization and mapping at the level of objects. In *CVPR*, 2013.

[58] Andreas Schindler, Georg Maier, and Florian Janda. Generation of high precision digital maps using circular arc splines. In *Intelligent Vehicles Symposium (IV)*, 2012.

[59] Grant Schindler, Panchapagesan Krishnamurthy, and Frank Dellaert. Line-based structure from motion for urban environments. In *3DPVT*, pages 846–853, 2006.

[60] Turgay Senlet and Ahmed Elgammal. A framework for global vehicle localization using stereo images and satellite and road maps. In *ICCV Workshops*, 2011.

[61] Noah Snavely. Bundler.

[62] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *CVPR*, 2014.

[63] StreetEasy.com. http://streeteasy.com/.

[64] Yuxiang Sun, Ming Liu, and Max Q-H Meng. Wifi signal strength-based robot indoor localization. In *Information and Automation*, 2014.

[65] Richard Szeliski. Where am i? In *ICCV*, 2005.

[66] Aparna Taneja, Luca Ballan, and Marc Pollefeys. Never get lost again: vision based nav. using street images. In *ACCV*, 2014.

[67] J-P Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *ICCV*, 2009.

[68] Stephen P Tarzia et al. Indoor localization without infrastructure using the acoustic background spectrum. In *Mobile Sys. App. and Srv.*, 2011.

[69] Mostafa Uddin and Tamer Nadeem. SpyLoc: a light weight localization system for smartphones. In *Sensing, Communication, and Networking*, 2014.

[70] D Van Opdenbosch et al. Camera-based indoor positioning using scalable streaming of compressed binary image signitures. In *ICIP*, 2014.

[71] RG von Gioi, J Jakubowicz, J-M Morel, and G Randall. Lsd: A fast line segment detector with a false detection control. *PAMI*, 32(4):722–732, 2010.

[72] Anh Vu, Jay A Farrell, and Matthew Barth. Centimeter-accuracy smoothed vehicle trajectory estimation. *Intelligent Transportation Systems Magazine, IEEE*, 5(4):121–135, 2013.

[73] Changchang Wu. VisualSFM.

[74] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *ICCV*, 2013.

[75] Paul A Zandbergen and Sean J Barbeau. Positional accuracy of assisted gps data from high-sensitivity gps-enabled mobile phones. *Journal of Navigation*, 64(03):381–399, 2011.

[76] C. Zhang, K. Subbu, J. Luo, and J. Wu. GROPING: Geomagnetism and crowdsensing powered indoor navigation. *IEEE Trans. on Mobile Computing*, 14(2):387400, 2015.

[77] Yinda Zhang, Shuran Song, Ping Tan, and Jianxiong Xiao. PanoContext: A whole-room 3d context model for panoramic scene understanding. In *ECCV*, 2014.

[78] Zhengyou Zhang. Iterative point matching for registration of free-form curves and surfaces. *IJCV*, 13(2):119–152, 1994.

[79] Baojiang Zhong, Dongsheng Xu, and Jiwen Yang. Vertical corner line detection on buildings in quasi-manhattan world. In *ICIP*, 2013.

[80] Zhiwei Zhu et al. High-precision localization using visual landmarks fused with range data. In *CVPR*, 2011.